ANALYZING LIFE-LOGGING IMAGE SEQUENCES

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Mohammad Moghimi Najafabadi

January 2017

ANALYZING LIFE-LOGGING IMAGE SEQUENCES

Mohammad Moghimi Najafabadi, Ph.D.

Cornell University 2017

With the abundance of ubiquitous cameras, it has become easier people take pictures of everything and everywhere. People take pictures of their possessions, interesting subjects and the places they visit. There is a class of passive cameras that let people be present in the moment while recording the situation. This act is called Visual Life-logging. Cheap cameras, storage devices and recent advancement in Computer Vision has created a unique experience. Life-logging has many applications besides its unique life recording perspective one of which is health monitoring. A camera can augment other health monitoring systems such as motion, blood pressure and blood sugar levels. We design algorithms to analyze life-logging image sequences to facilitate public health research. Our approach to the analysis is threefold: unsupervised, supervised, human-in-the-loop.

We designed an algorithm to extract regions of interest from image sequences based on their occurrences in different scenes. We used the histogram of gradients (HOG) feature and applied a repetitive classification discriminatory approach to finding patches that only appear in a scene but not other scenes. Using our method, we can discover objects such as a monitor in an office setting or bike handles in a biking scene in an unsupervised manner.

The next step is to analyze the data in a supervised fashion. After carefully designing a set of labels appropriate for the public health research which includes posture, activities, scenes and social settings, our team has manually annotated the data with these labels, and we implemented visual classification algorithms to classify images us-

ing these tags. Our methods include state-of-the-art pre-deep learning models as well as deep convolutional neural networks. We extend the CNN with spatial, temporal and model-level bagging and model-level boosting. Unique characteristics of life-logging image sequences require a custom model to leverage these aspects such as temporal coherence and correlation of images of each person.

The annotation of the dataset consisting of millions of images is a cumbersome task. It requires extensive time, money and resources. In this thesis, we present the foundational tools to efficiently annotate the image sequence by leveraging the previously labeled data to minimize annotation time and increase the accuracy. Our experiments show a significant decrease in annotation time.

BIOGRAPHICAL SKETCH

Mohammad Moghimi Najafabadi was born on January 24th, 1985 in Tehran, Iran where he spent most of his life. He studied Computer Engineering and received BSc from University of Tehran and 2007 and MSc from Sharif University of Technology in 2010. He moved to the United States and started his PhD studies in UC San Diego before moving to Ithaca to continue his PhD. He expects to get a PhD in Computer Science with a minor in Electrical and Computer Engineering from Cornell University in January 2017.

I dedicate this thesis to my parents Narjes and Jamshid and my lovely wife Elham without whom this PhD would not be possible.

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the help and support of a large number of people.

First, I would like to thank for PhD advisor Prof. Serge Belongie for his immense support throughout the duration of my PhD. His encouragement to forge my own path and his focus on the big picture have been a source of inspiration and have been crucial for my work. Working with him has helped my understand advanced Computer Vision methods from both academic and real-world industrial perspectives which I am very grateful for. I also thank other members of my dissertation committee Prof. Noah Snavely and Prof. Tsuhan Chen for their feedback on this thesis. I would also like to thank my previous committee members at the University of California, San Diego for the comments in the beginning of PhD Prof. David Kriegman, Prof. Gary Cottrell, and Prof. Sanjoy Dasgupta.

I must thank members of SE3 lab for the ever going discussions and fruitful feedback during my PhD. Thanks to Hani Altwaijry, Tsung-Yi Lin, Yin Cui, Michael Wilber and Andreas Veit. I must also thank SO3 colleagues Zachary Murez, Oscar Beijbom, Catherine Wah, Steve Branson, Kai Wang, Iljung Kwak who have inspired me to proceed with my research. I should also thank members of graphics and vision lab at Cornell: Prof. Kavita Bala, Omid Poursaeed, Kevin Matzen, Kyle Wilson, Paul Upchurch, Sean Bell and Scott Wehrwein for their help in my research.

My sincere thanks to the iWatch team with whom this work would not be possible. Their help in shaping my research, providing data and support were necessary for my PhD work. I would like to thank Prof. Jacqueline Kerr, Prof. Simon J. Marshall, Katherine Ellis, Suneeta Godboleh, Eileen Johnson, Grant Fraley, Jacqueline Chen and Katie Crist.

Finally, I must thank my parents for aspiring me to continually learn from those early

days and my dear wife Elham who has been with me day and night during my PhD. This PhD was not possible without her sacrifices. I also like to thank our bundle of joy, our 2-year-old son, Taha for giving me the motivation to work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

With the abundance of ubiquitous cameras, it has become easier for people take pictures of everything and everywhere. People take pictures of their possessions, interesting subjects and the places they visit. They take pictures and selfies to record a moment in their lives to re-live that moment or to share it with others. There is a class of passive cameras that let people be at the moment while recording the situation. This act is called passive visual life-logging. Cheap cameras, affordable storage devices and recent advancements in Computer Vision have created a possibility of a new experience.

Visual Life-logging has many applications. These applications span a spectrum of real-time to offline applications.

1.2 Applications

real-time applications. This category of applications includes interactive tools. It is possible to build a real-time application if a device could analyze or transmit the data for remote processing as it is being recorded and provide real-time feedback to the user. There are many possibilities of real-time applications based on these kinds of devices. One example could be physical activity intervention. It is known that long sitting periods hurt the body. A device could warn the user to move after detecting a long of sitting posture. Another example is a device that predicts certain situations and warns the user. It is helpful to remind some people that some activities are harmful to their bodies.

When a subject approaches a smoke shop, a warning can give the possibility of thinking twice before buying a cigarette. Or an interactive life-logging device could give driving suggestions. These kinds of warnings or recommendations could significantly impact peoples' lives. These devices could also help augment human vision or memory. They can greatly help blind or semi-blind people to see what is around. They can also help people remember. This application is not only helpful for people with memory-related diseases such as Alzheimer's, but it is also useful for others without such deficiencies. Have you ever forgot someone's name? Another example is visual-based reminders. Reminders typically work based on time or location. Imagine, if you could ask your assistant to remind you to buy an item at your grocery store when you are in front of that shelf.

offline applications. The other set of applications is offline applications. The idea is that the device records the data and later sends them to a server for processing. There is a suite of possible applications in this category. The data could be used to summarize for a variety of purposes such as viewing, medical or professional use. You may want to get an automatically generated video summary of your day when you travel. Summarization of a first person camera of a surgeon doing a surgery needs to look very differently. Another kind of summarization is for health monitoring. We primarily focus on this group of applications in this thesis.

health monitoring. There are endless possibilities for health monitoring using body-worn cameras. We specifically focus on physical activity analysis of life-logging images for public health research. This thesis builds the foundation of acquiring data to make medical research conclusions. Let's say; a medical hypothesis is if "long sitting periods increase the risk of colon cancer?". The best way to test the hypothesis is to measure accurate metrics based on real people outside of the clinics, in their homes,

work and in everywhere in between. In this thesis, we make the necessary tools to make the data acquisition easier.

As we mentioned, our focus is analyzing physical activities including human activities and postures. We are interested in labeling visual life-logging image sets with activity and posture labels. This metadata will then used in medical (public health) research.

1.3 Contributions

The goal of this thesis is to build tools to process life-logging imagery. We define visual life-logging data as image sequences captured by wearable cameras. There is a spectrum of different frequencies at which the images are recorded. One end of this spectrum is high-frequency image sequences also known as videos while the opposite end contains very sporadic images as few as a dozen per day. Any fixed frequency or dynamic life-logging camera lies somewhere in this spectrum. There are a variety of use cases that require different technologies that are suitable for each use case. Chapter. 2 reviews different wearable cameras. For example, Google Glass is designed for interactive applications with a slick and light design. While the camera sensor and lens of Google Glass are not the best of their kinds, the trade off in design makes it possible to wear it on the face. GoPro, made primarily for filming sports, is capable of surviving many situations that damage most cameras. We chose SenseCam, a chest-worn camera made by Microsoft Research. The reasons for choosing this device were its form factor, durability, battery, storage, price, and its wide lens. It captures images at 0.05 to 0.03 frames per sec (fps) resulting in a few thousand images per day. We have gathered dataset of more than a thousand days from hundreds of subjects.

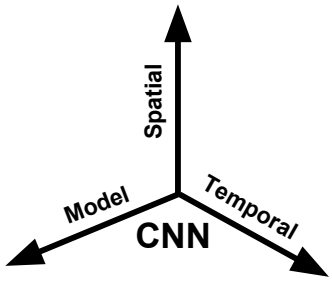


Figure 1.1: Classification Contributions.

After gathering, cleaning and managing the data, we approach the data analysis from three different perspectives: unsupervised, supervised and human in the loop. First, we designed tools to extract information from images sequences without manual labels automatically. We, developed an algorithm to find regions of interest that are unique to a given scene. We defined characteristics of such regions and designed a method to extract them in an unsupervised fashion. Then, we built a model to learn different position and activity categories. We started with traditional region based methods such as SIFT, Pyramid Match Kernel and Fisher Kernels, and holistic features such as color histograms and GIST. We show that our Deep Convolutional Neural Network based method outperforms traditional features. Furthermore, we extend deep learning models in three different ways: stacking spatial, temporal and model information. Fig. 1.1 demonstrates these different aspects.

We show that while convolutional networks are powerful, they can be augmented with spatial and temporal information. We gathered features from various windows in the scale and spatial space as well as information from consecutive frames to capture the temporal aspect. This multidimensional, spatiotemporal information is obtained by concatenating multiple feature representations.

In addition to that, we took it one step further by utilizing multiple models. We first showed that we could improve performance by averaging predictions from various

models and then extended it to a boosting framework where each model is trained to complement a set of convolutional networks that are trained one at a time.

1.4 Organization

This thesis is a collection of independent but related chapters. Thus each chapter can be regarded as an isolated and complete research contribution. It is composed of 5 main chapters.

Chapter. 2 reviews different works in life-logging and wearable computing. Chapter. 3 discusses our unsupervised algorithm to find discriminative regions. Chapter. 5 explains our classification solution based on image sequences. Chapter. 4 presents a similar approach on another wearable video dataset we collected. Finally, Chapter. 6 demonstrates our boosting algorithm which shows the state-of-the-art result on several public computer vision datasets.

CHAPTER 2

EGOCENTRIC COMPUTER VISION: A SURVEY

2.1 Abstract

In recent years, there has been a surge of interest in wearable cameras in the consumer electronics industry. However, there is a need for efficient and accurate image analysis techniques for processing images and videos captured by these wearable devices. In this report, we categorize and explain recently published research in the field of Egocentric Computer Vision. We also list successful wearable cameras, their features and the available computer vision datasets captured by these cameras. After discussing basic building blocks that researchers have developed, we review different applications of Egocentric Computer Vision such as life logging, summarization, health-related applications, activity and place recognition. We conclude by considering potential directions for future research.

2.2 Introduction

With recent advances in computing technology, more and more cameras are being built and used by people every day including analog cameras, digital cameras, mobile devices and wearable cameras. The challenge is how to manage and process the vast amount of data that is being captured by these devices. There has been an extensive amount of research on analyzing images from digital cameras (e.g. web images), fixed installed cameras (e.g. surveillance cameras), and mobile devices (e.g. smartphones). However, there is a gap in research concerning wearable cameras such as Google Glass, SenseCam, Memoto or Looxcie.

These wearable cameras are usually equipped with various sensors such as GPS, accelerometer, light or infrared sensors. These wearable cameras, with their additional sensors, capture a different view of the world that is not possible with other types of cameras. The new perspective helps the wearable cameras perceive the environment the way humans do. Furthermore, they allow many applications that are impossible without these kinds of cameras. Egocentric Computer Vision is an emerging research area that focuses on the visual analysis of wearable cameras with a variety of applications such as studying human perception and human assistance.

There is not a single definition for Egocentric Computer Vision. First of all, the ego can be used to refer to a human individual, a robot, a vehicle [40] or other things depending on the application. Wearable cameras such as Google Glass or Memoto fit in the human-centric vision category while cameras used for automated car navigation or robot localization fit in vehicle-centric or robot-centric vision categories. In this survey, we focus on the applications of the Computer Vision for human-centric cameras. Secondly, researchers use different names for this research area including First Person Vision and Wearable Computer Vision. Among these names, Egocentric Computer Vision is the most popular term due to two recent workshops in this area in CVPR 2009 and CVPR 2012.

Egocentric Computer Vision has many applications. These applications include assistive technology, health care and monitoring, navigation, early hazard detection and life logging. In most cases, to get better performance, researchers employ other wearable sensors as well. Thus, egocentric computer vision should combine with other research areas such as Pervasive, Ubiquitous, Wearable and Mobile Computing. Egocentric Computer Vision has emerged only recently due to limitations in the image capturing devices such as power usage, storage, and size. However, researchers have been envi-

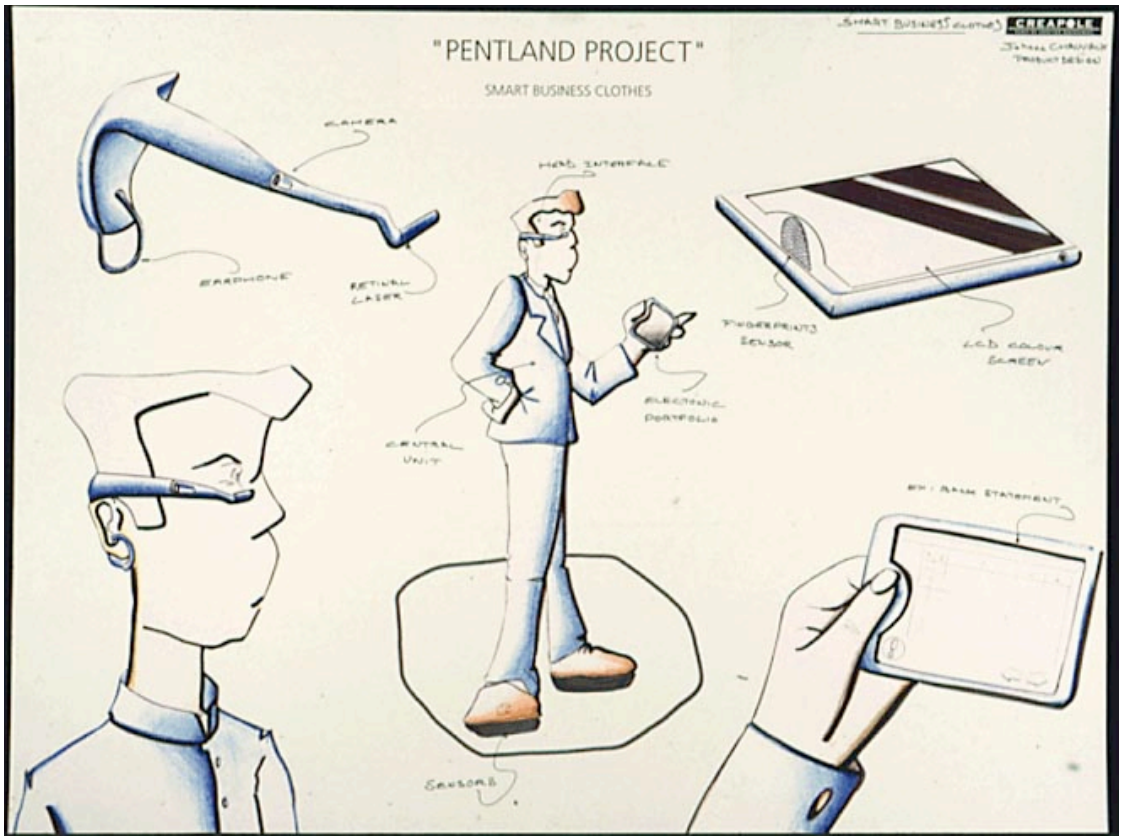


Figure 2.1: Alex Pentland's vision.

sioning using wearable cameras for many years. Fig. 2.1 shows a sketch Alex Pentland's ideas from 1998. In the past two decades, researchers have been working to develop new devices and algorithms to change how humans interact with the world, and recently there have been significant advances towards that dream.

In this survey, we discuss different aspects of wearable cameras. First, in Sec. 2.3, we explain foundational Computer Vision methods for processing egocentric videos and images. Then, in Sec. 2.4 and Sec. 2.5, we note different wearable cameras and publicly available datasets . Sec. 3.7 considers applications of wearable cameras, and finally Sec. 3.9 concludes with potential future directions.

2.3 Methods

2.3.1 Gaze

Human gaze is an important source of information for egocentric applications [30] [140] [127] [95]. Human gaze, which indicates the visual attention of the first person, helps algorithms analyze the surrounding environment. The reason is that people are usually working with objects that they are looking at. This means that first person vision systems should specifically focus on those areas in images. The best way of estimating gaze is with special hardware. It can be estimated using either visual or non-visual sensors.

The most common way is to use cameras pointing at the wearer's eyes. In this way, the sensing device can track the eyes' movements to estimate where the wearer is looking. In 2010, Hansen et al. comprehensively reviewed of gaze estimation methods [50]. Gaze estimation from back-facing cameras consists of two simpler sub-problems: eye detection and eye tracking.

There are many works on eye detection. We divide them into three broad categories [50]: shape-based approaches, feature based shape methods, and hybrid approaches. The shape-based methods assume a generic shape for the eyes and sometimes the eyelids. This shape model is usually a simple elliptical form or a more complex shape model. Using a voting based method, possible eye locations are detected and used for further processing. Feature-based shape methods extract features related to different parts of the eye such as its corners or the pupil. Intensity changes or oriented filter responses are usually used in these types of methods. The last category of eye detection methods includes hybrid methods where both shape features and local features are used

together to enhance eye detection.

Eye tracking and detection are two closely related problems that have to be solved together for better performance. The first step is to detect eyes. The next step is tracking which is done with methods such as mean shift or Kalman filters. Eye tracks may drift from the eye's correct location. However, the temporal information helps with reducing the noise.

Non-visual signals are another way of estimating visual attention. Researchers have looked at different brain signals for this purpose. Electrooculography (EOG) is a technique for measuring the resting potential of the retina. Specifically, Bulling et al. used EOG [9] and designed a special head covers to record EOG signals with multiple sensors. After analyzing EOG signals, they extracted features from the sensed voltage to represent whether the person is moving his/her eyes to the left, right, down or up. A sequence of noisy micro movements was used to classify different reading actions. They designed a particular string matching algorithm to classify reading when the reader is walking, standing, sitting or riding. To get the best performance, they used a Hidden Markov Model (HMM) based temporal model to represent the transitions between sub-movements. Their system was able to classify different reading activities with a recognition rate of 80.2%.

Image based (sensor-less) gaze prediction is another trend of research in image processing and neuroscience fields. There are an extensive number of research papers from neuroscientists in modeling human attention. Laurent Itti's works are among the most famous in this area. He came up with a computational model based on image features [57]. Machine learning researchers also attempted to create computational models for human gaze. Judd et al. [60] recorded eye tracking data for over 1000 images and trained a linear support vector regression model to predict the human eye gaze.

2.3.2 RFID

Radio-frequency identification (RFID) is a technique that can be used to find the presence and distance of objects by attaching RFID tags to objects. Recent advances in RFID technology have made RFID tags cheap and affordable. To get RF signals, the human wears a RFID reader, which can be in different forms such as a bracelet or a necklace. The reader receives RFID signals from the objects while the wearer is working with them. In fact, the RFID reader measures the distance to different objects and these distances can be used for many applications. Here we mention research works that use RFID for egocentric activity recognition.

Patterson et al. in [99] and [101] created a system that only uses RFID tags to detect different activities such as making oatmeal, making coffee, eating breakfast and clearing the table. They created a dataset where they annotated different labels for various activities and used an HMM approach to solving the classification problem.

Wu et al. made a system that can recognize 16 activities (such as boiling water, making tea, drinking juice and making a salad) using interactions with 33 objects (such as a water jug, kettle, teabag, and dressing) [138]. To get from noisy RFID signals to activities, they have used a deep belief net (DBN) to model the relationship between activities and sensor readings. Furthermore, they used a wearable camera and a bag of visual words (BOW) model to detect objects using the camera and added the visual detections to the belief net. They thus showed that it is beneficial to use both visual and RFID sensors.

Spriggs et al. in [125] used the RFID marker as well as IMU sensors (i.e. accelerometer and magnetometer) and an egocentric camera to detect 29 different short actions involved in making a brownie recipe. The activities include ‘open fridge’, ‘close fridge’,

'get fork', 'get eggs', 'walk to counter', etc. They attached RFID tags to some objects while it was not possible to attach RFID tags to other objects such as eggs or forks. The detection of such objects remained for Computer Vision algorithm that processes the video. They used an HMM and K-nearest neighbor classifier (KNN) that mixes the RFID reading and visual detection for the classification of the selected activities.

2.3.3 Object Recognition

Object Recognition is a well-established problem in Computer Vision with numerous application and methods. Object recognition methods can be divided into three main groups: generic category level recognition, specific category recognition, and object instance recognition. Generic Object Recognition methods can recognize objects of different categories such as cars, faces, airplanes, etc. However in specific category recognition are methods that are specialized for a specific category such as faces [134], cars and pedestrians [24]. Given a specific category, we can engineer visual features and object models to get the best performance for the given category such as Haar features for face detection. The third category is object instance recognition methods which are defined as searching for a specific instance of an object category.

There are many successful methods for object detection such as BOW, Spatial Pyramid Match Kernel (SPMK) [74], deformable part models [34] and Haar-based boosting methods. All of these methods are based on local features (e.g. SIFT [81]) that are extracted from local regions in images and use different methods to combine and relate features extracted from different parts of the objects. BoW models discard the geometry information and use histograms of quantized feature vectors (visual words) to encode an image. Later, a classifier is trained on training images to create object models using

the extracted histograms. SPMK pools the features from different parts of the image and encodes geometry of image regions to some extent. Finally, deformable part models explicitly model object parts and learn to detect object parts in a semi-supervised manner.

Boosting methods are another group of successful object recognition methods for fast rigid object detection such as face detection. The key factor in these methods is the ability to compute Haar features very fast which are the building blocks (i.e. weak classifier) for the boosting algorithm.

Recently, researchers have done object recognition for egocentric cameras. There are several factors that egocentric cameras benefit from. For example, gaze either in direct or indirect form helps find and segment the objects of interest. Here, indirect gaze estimation means using the center of the egocentric image as the gaze. The assumption is that head movements help the eyes fixate on the objects of interest so that they reside at the center on the eyesight. Hands also play a very important role in the human-object interaction. A reliable hand detector can help find objects that are later classified.

Schiele et al., in 1999, made an automatic system to detect objects with an egocentric camera using the histogram of local feature detectors [116]. Recently, Fathi et al. [32] created an object recognition system that uses hand models to find objects. They first built a panorama for the background and used it to segment the foreground. Based on the color and texture, they segmented hands vs other foreground regions which are later used for classification. Ren et al. [109] used a similar approach but instead used optical flow based foreground/background segmentation to find the foreground regions. They showed an improvement of using foreground segmentation on object recognition. Their system had the option not to choose the foreground segmentation for those objects which were static in the experiment.

2.3.4 Activity Recognition

Activity recognition is defined as recognizing different human actions from a video. It requires temporal information in addition to the appearance information. Laptev et al. performed much of the early work on unconstrained activity recognition [73] [72] [85]. The unconstrained setting means that the actor has not done the activity for the Computer Vision research purpose as opposed to constrained settings such as the KTH action dataset [117] in which actors are doing different actions in front of a fixed camera with a very constant background. Laptev et al. proposed spatio-temporal features that not only look at scene appearance but also process movement by considering gradients in time and space to encode actions. In their work, they applied their algorithms on real Hollywood movies in a supervised learning setting.

One of the real problems in activity recognition is that there is no single definition of when an activity starts and ends. Instead, activity models should be able to take into account that the input time window might be smaller or larger than the activity itself. Furthermore, there are always some activities happening and that are correlated. Picking the right set of activities is a challenge to key a successful activity recognition system.

2.3.5 Hand Detection and Tracking

Hand detection and tracking are essential parts of successful egocentric computer vision systems. It is very typical for first person view cameras to capture images of hands in great detail. However, depending on the placement of the wearable camera (e.g. chest, head, ear, etc), hands appear differently. A good view of hands helps to automatically detect and track hands which are useful in many applications such as hand gesture recognition or finding objects of interest. Here we mention a few successful works on hand

detection and tracking.

EyeWatchMe [127] is one of the papers that is targeted at the tracking of hands. The authors used a two-stage approach. First, they used a 2D model with 9 degrees of freedom (DOF) to localize the hands. Then a more detailed 3D hand model with 27 DOF is adopted. EyeWatchMe is also capable of tracking other objects, and they analyzed human gaze and tried to find correlations with the object tracking system when certain actions such as pouring hot water into a cup are being done.

Mayol and Murray [87] used a probabilistic method to segment hands based on skin color and skin area. To reduce noise, they used morphological operations to find large regions of possible skin areas. Detecting hands in each frame results in a trajectory for hands which is used to classify different hand activities such as using a keyboard, handling tennis ball, using calculator and resting.

Fathi et al. [32] designed a different approach for hand detection. The first step in their method is foreground-background segmentation. The assumption for background subtraction is that the background is static in the world coordinate frame and everything that moves with respect to the background is foreground. That means some objects that do not move throughout the video are part of the background. First, the larger background is modeled and then foreground regions are judged based on it. The foreground regions consist of hands and other objects. They assumed that the hands cover most of the foreground areas. It is a correct assumption based on the data that they are using. In their object manipulation dataset, hands are visible most of the time. Thus, by comparing a color histogram of the whole foreground region and a given super-pixel, one can determine whether the super-pixel is part of the hands or another object. Additionally, because of the nature of egocentric camera viewpoint, it is possible to distinguish between the two hands based on their locations in the video.

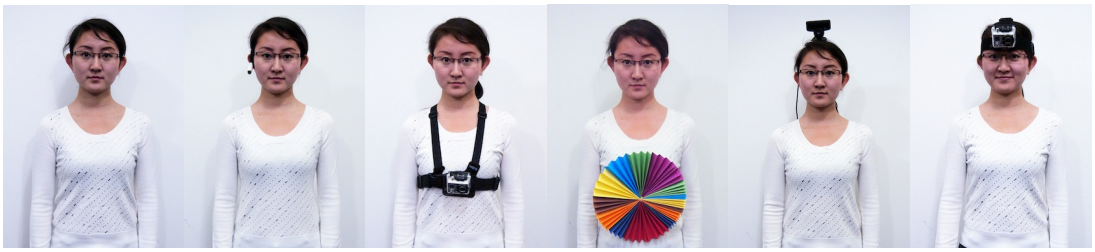


Figure 2.2: Different device placements [52].

2.4 Devices

To do research in egocentric vision, one should be familiar with the wearable cameras used to capture images, record videos or record other wearable sensors. These cameras are designed both in academic institutions (e.g. [20], [51], [103]) and industry. Wearable cameras are designed for different purposes such as life logging, recording first person sport videos or studying human attention. Depending on their goals, these cameras are equipped with other sensors such as accelerometer, temperature/light sensor and/or GPS. Other than availability and quality of different sensors in these devices, privacy is an issue. The privacy and obtrusiveness of the wearable device may limit possible applications. Hayden et al. have looked at the trade-off between accuracy of face detection versus obtrusiveness of the wearable vision device [52]. Fig. 2.2 shows a study of Hayden et al. Based on their survey, the smaller the device is, the more acceptable it would be. In this section we review popular devices that can be used by researchers to create egocentric applications.

FPV. Devyver et al., in [20], discussed the implementation of their first-person camera that is mounted on the right side of a frame of glasses. They employed two cameras: a forward camera with the same view as the wearer and a backward camera recording the position of the eyes. The latter is used for gaze tracking and is considered the best way

of estimating human gaze. They demonstrated results of their gaze tracking algorithm which is done offline after transferring the videos to a PC.

WearCam. Piccardi et al. [103] designed a wearable camera (called WearCam) that is installed on the forehead. They followed a similar path to [20] in designing the gaze tracking system which is a camera looking at the eyes. The difference was that they used a mirror and a forward camera to do this task. With this simple idea, they could design a very cheap camera that can track eyes reasonably well in most of the conditions.

Google Glass. According to Wikipedia[1], Google Glass is set to be available for consumers by early 2014. Based on Google's reputation and the availability of the API, this device is going to one of the leading platforms for egocentric applications. It is a glasses frame with a battery, a display, a camera and a set of other sensors. It can connect to a cellphone via Bluetooth for extra processing and internet access.

GoPro. GoPro[2] is targeted at adventure video/photography. The company has some different cameras designed for different situations. They are designed for filming sports such as surfing, snowboarding, skydiving and others. Thus special mounting features as well as hard and strong bodies, are the leading character of these cameras. HERO 3, the latest 2012 version, captures up to 240 frames per second with 848×480 resolution.

Tobii. Tobii Technology Inc has many eye tracking systems including eye tracking glasses[3]. The eye tracking glasses capture 640×480 resolution video at 30 frames per second. It is equipped with a microphone and uses infra-red to enhance eye tracking.

SMI Eye Tracking. SMI eye tracking glasses are another head-mounted device that tracks eyes and registers it with the video that is being recorded. It records HD video

[1] http://en.wikipedia.org/wiki/Project_Glass
[2] http://www.gopro.com/
[3] http://www.tobii.com/en/eye-tracking-research/global/products/hardware/tobii-glasses-eye-tracker/

at 24 frames per second. SMI eye tracking has a very stable eye tracking system and works even with contact lenses.

SenseCam. SenseCam was developed by Microsoft Research [54] and it has been licensed to Vicon. It is designed for a variety of medical applications and is now available commercially as the Vicon Revue. It is a chest mounted camera which takes VGA images (640×480) at a predefined interval and can store up to 30000 images on its internal storage. It is equipped with other sensors such as temperature, light sensor, accelerometer and has ability to connect to an external GPS sensor.

Memoto. Memoto[4] is designed by Memoto AB, a Swedish startup company. It is a small wearable device that takes pictures with a fixed frequency. The good battery of this device lasts for about two days which is equivalent to 4000 images. One of the advantages of this device is the cloud storage service developed by Memoto.

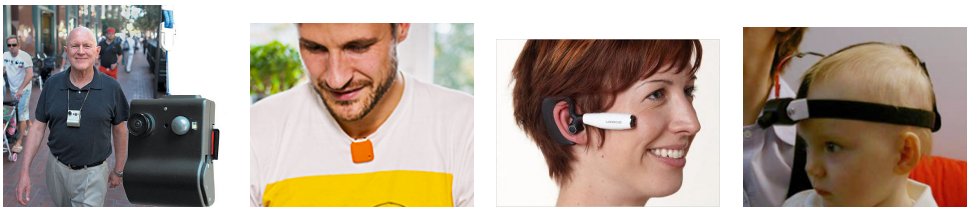
Looxcie. Looxcie[5] is designed by Looxcie Inc, an American startup company. It is a small and light wearable device that is easy to wear on an ear. It also looks and functions like a Bluetooth headset. In the low-resolution mode, 320×240, the camera records up to 5 hours of continuous video which is considerably good for a wearable camera. It also records higher quality videos and has a version that records high-quality HD videos as well.

2.5 Datasets

In this section, we list all of the public first person vision datasets used by the Computer Vision community.

[4] http://www.memoto.com/
[5] http://www.looxcie.com/

(a) Google Glasses (b) GoPro (c) SMI Eye Tracking Glasses (d) Tobii Eye Tracking Glasses



(e) SenseCam (f) Memoto (g) Looxcie (h) WearCam

Figure 2.3: Popular devices for egocentric vision.

Figure 2.4: Sample frames from GTEA Gaze dataset.

2.5.1 GTEA Gaze

GTEA Gaze[6] dataset was gathered in 2011 [32]. It consists of 17 different sequences for videos performed by 14 different subjects. The sequences are all about kitchen works and include 7 activities and 16 kinds of objects. They provide gaze information for each video frame estimated by the camera as well as labels for the periods when the actions are happening. Fig. 2.4 shows sample frames from GTEA Gaze dataset.

[6]http://www.cc.gatech.edu/ afathi3/GTEA/

Figure 2.5: Sample frames from GTEA Gaze+.

2.5.2 GTEA Gaze+

GTEA Gaze+ [7] dataset which is a successor to the GTEA Gaze dataset, is collected as a part of Georgia Tech's AwareHome project [30]. It consists of 7 meal preparations performed by ten subjects. They used SMI eye tracking glasses which provides better resolution and framerate as compared to Tobii glasses. The activities include preparation of American Breakfast, Pizza, Snack, Greek Salad, Pasta Salad, Turkey Sandwich and Cheese Burger. Aiming to create a better dataset, they also annotated 100 objects as opposed to 16 objects in the previous dataset.

2.5.3 UCI ADL

UCI Activities of Daily Life (ADL) dataset [8] is collected by Pirsiavash et al. [104] from UC Irvine. This dataset is collected using a GoPro camera capturing high definition quality videos (i.e. 1280×960) with a wide 170-degree viewing angle. This dataset consists of 18 daily activities such as watching TV, combing hair, doing the laundry and making tea. 20 people participated in the data collection to create a 10 hour combined video. This dataset is both used for object recognition, and activity recognition as object labels are annotated as well. Fig. 2.6 shows one frame with its object annotations.

[7] http://www.cc.gatech.edu/ afathi3/GTEA_Gaze_Website/
[8] http://deepthought.ics.uci.edu/ADLdataset/adl.html

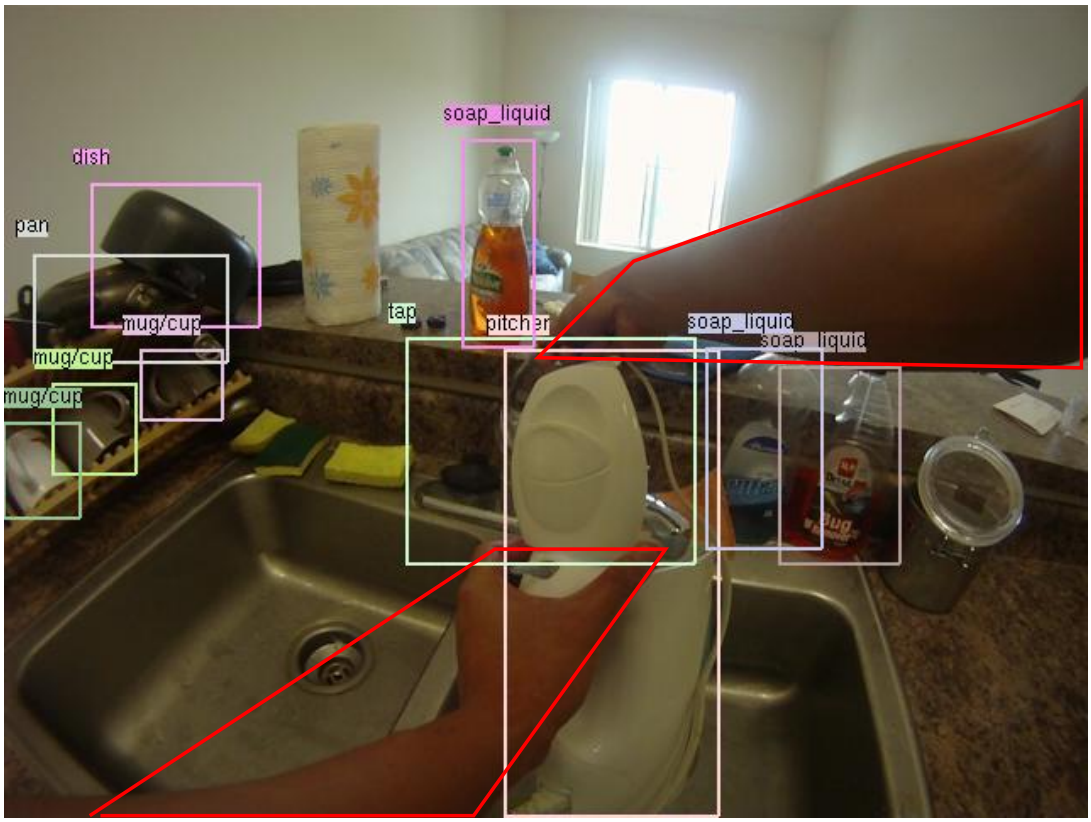


Figure 2.6: A sample frame with object annotations from UCI Activities of Daily Life (ADL) dataset.

2.5.4 First-Person Social Interactions Dataset

First-Person Social Interactions Dataset[9] is composed of 8 days worth of video recorded by six people who spent a day in Disney World Resort in Orlando, FL [31]. Since the cameras are worn at the same time by multiple people, interesting problems can be solved using this dataset. Fathi et al. made a system to detect a social interaction between people using multiple cameras in the same scene.

[9]http://www.cc.gatech.edu/ afathi3/Disney/

2.6 Applications

2.6.1 Activity Recognition

Activity recognition is one of the most important applications of Egocentric Computer Vision. Having a better viewpoint gives an advantage to egocentric cameras to capture the human action compared to third person cameras for many activities. This viewpoint difference makes some techniques fail while helping other methods. One crucial assumption in third person cameras (e.g. surveillance cameras) is that the background is not changing which helps to segment the foreground regions for later processes. On the other hand, egocentric viewpoint helps in other cases such hand detection, tracking and hand gesture recognition.

Human activities are very diverse, structured and complex. Researchers have focused on different subsets of human activities. Recent works can be categorized into three categories: object-based, motion based and gaze based activity recognition.

Object Based Activity Recognition. Most of the egocentric activity recognition papers fall in this category. Object-based activities are defined as activities that involve the interaction of a human with some object(s). In some cases, depending on the activities, the objects solely define the activity. For example, in making a peanut butter jelly sandwich, it might be enough to detect relevant objects such as bread, peanut butter and jelly and knife. Sometimes, the order in which the object appear is important to detect the activities. For example, to detect whether the lid of a peanut butter jar is being closed or opened, it is important to detect the lid before or after the closing or opening of the lid. Additionally, since the state of these objects changes during the activity, it is beneficial for the Computer Vision system to be able to detect object/environment state changes.

For example, peanut butter spread on a piece of bread is an example of change of state for bread.

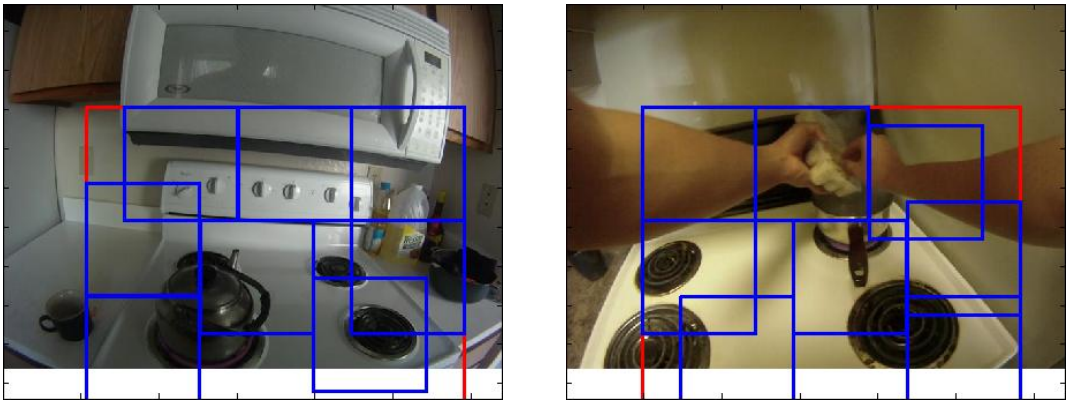
Research on object based activity recognition has mostly been done in indoor environments where the number of objects is limited. Pirsiavash et al. [104] did an indoor experiment in which they asked 20 people to do 18 different known activities in their homes including washing hand/face, combing hair, laundry, watching TV, making tea and drinking water bottle. Their main assumption is that activities can be represented by a set of objects and their appearance in time and space. For example, making tea requires first boiling water, pouring it into a cup and putting a tea bag in the cup. This activity includes three objects that are a teapot, tea bag, and cup. It is also important where and when they appear in the video.

Pirsiavash et al. built their work on an object-based representation of activities. Their representation is a temporal pyramid of objects which is different than spatio-temporal pyramids such as [13]. In this model, they define a score function for every location, size $p = (x, y, s)$, frame number (t) and object (i). They used a deformable part [34] model approach to compute the score function. Additionally, f_i^t is defined as the maximum score for a given object in a frame.

$$\text{score}_i^t(p) \in [0, 1] \tag{2.1}$$

$$f_i^t = \max_p \text{score}_i^t(p) \tag{2.2}$$

Using the f values, they construct a temporal pyramid in the following way, where there are bins of different sizes that build a pyramid. x, the feature vector, represents a



(a) passive stove (b) active stove

Figure 2.7: Passive vs active stove [104].

coarse-to-fine histogram for a set of frames. Then these feature vectors are then classified using a linear SVM classifier.

$$x_i^{j,k} = \frac{2^{j-1}}{|T|} \sum_{t \in T^{j,k}} f_i^t \qquad \forall k \in \{1, ..., 2^j\} \tag{2.3}$$

Additionally, they introduced the notion of active objects. It is a fact that objects have different appearances when they are being interacted with. Also, because of the nature of the egocentric video, active objects appear in certain locations of the image. For example, a model for an active stove (Fig. 2.7) should consider the occlusion of hands and the high probability of the stove location at the bottom of the image. In their method, they trained separate object detectors for active objects and added the location of the object to the feature vector to model the active object location as well.

Motion Based Activity Recognition. Motion is another source of information for activity classification. Motion is important in almost every activity, and there are certain activities that are defined just based on movement. The source of these movements can

be the first person, his hands or other objects. First person movement activities include physical activities e.g. running or walking and activities that are defined by changes in the whole scene such as driving a car [66] [77]. The second category, hand movements, include different hand gestures such hand gesture recognition paper by Hanheide et al. [49]. The third category models the movement of hands when interacting with different objects [47] [48].

Kitani et al. [66] used a helmet-mounted GoPro camera to capture videos from rigorous sports scenes. They used a histogram of motion to describe the activity. The histogram consists of motion direction, amplitude, and variance from a set of sparse motion vectors. The noise in the motion vectors are removed by applying a planar homography model between consecutive frames, and they used RANSAC to find the best transformation. Using a probabilistic model, they designed an unsupervised activity discovery system which they later combined with specific labels for each sport to make a supervised classification. They defined about 10-20 ego-actions for each sport. For example, surfing includes underwater swimming, hitting waves or light paddling. They showed that their method is more accurate compared to other unsupervised clustering algorithms [77], HMM and still image activity recognition.

Hanheide et. al [49] made a system for tracking hand trajectories. They designed a kernel based tracker to track hand movements. They showed classification results for five categories of pour left, pour right, move left, move right and shake.

Gaze Based Activity Recognition. Some cameras such as Tobii or SMI are equipped with specific hardware that facilitate the gaze estimation process. There are two ways that gaze can help the activity recognition process: direct and indirect. There are some activities such as reading a newspaper or watching TV that are directly related to gaze while gaze can help in the recognition of other activities such as making a

sandwich by highlighting the important regions in the video.

Ogaki et al. [95] designed a system to directly use gaze for the set of eye related activities including reading, watching a video, writing, copying text using two screens and web browsing. First, they process the two-dimensional gaze signal to remove noise before extracting features from the signal. They later combined it with the global motion of the video which represents head movements. The concatenated features are clustered to make a codebook which later is used for classification.

Fathi et al. [30] used gaze to find important objects in the scene which they used for activity recognition. They combined the gaze information with their previous paper [32] to enhance the foreground segmentation.

2.6.2 Health-related Applications

Generally, there are many health related applications for wearable computing and special sensors including measuring blood pressure, sugar level, physical activity, air pollution and so on. In this section, we specifically talk about applications related to Alzheimer's disease and autism spectrum disorder.

Autism

Autism Spectrum Disorder (ASD) is a developmental disorder that touches 1 in every 160 children. It presents itself in impairments in social interaction, communication and stereotypical and repetitive behaviors. One of the symptoms of children with autism is reduced gaze towards social stimuli. Computer Vision researchers have built different systems to study gaze in children to detect Autism in its early stages. Since it is not

comfortable for children to wear gaze-tracking glasses, researchers used other methods to estimate their gaze. Noris et al. used a camera with a backward facing mirror to record eye movements [92]. Another method that Ye et al. used was not to equip the child with any devices and use face recognition algorithm to estimate the gaze [140] from a second person point of view.

Noris et al. [92] used a camera called Wearcam(Fig. 2.3). WearCam has two cameras and can record video while estimating gaze simultaneously. Their empirical results support the phenomenon of gaze downcast in autism. Their experiments verified the brain studies that explain the gaze downcast as a response to sensory overload coming from a hypersensitivity to visual stimuli.

Alternatively, Ye et al. [140] used a pair of glasses that is worn by an adult who interacts with the child. Their goal was to find moments of mutual gaze. They achieved this by finding the orientation of the child's face and the gaze direction of the child from the video. Then, they matched this when the adult's gaze is on the child's face. Their paper used two commercial tools for gaze tracking (SMI eye tracking glasses) and face orientation (OKAO Vision library).

Memory Aid

There exist some complex brain damages such Alzheimer's disease and Limbic Encephalitis that makes it hard for the patients to recall past experiences. Researchers have proposed different ways to help the patient recall events from his/her long term memory. One of these approaches is recording images of daily life to help the patient remember more details. This is one of the reasons for the design of SenseCam [54]. It is a chest mounted wearable camera that periodically and passively takes pictures and stores them

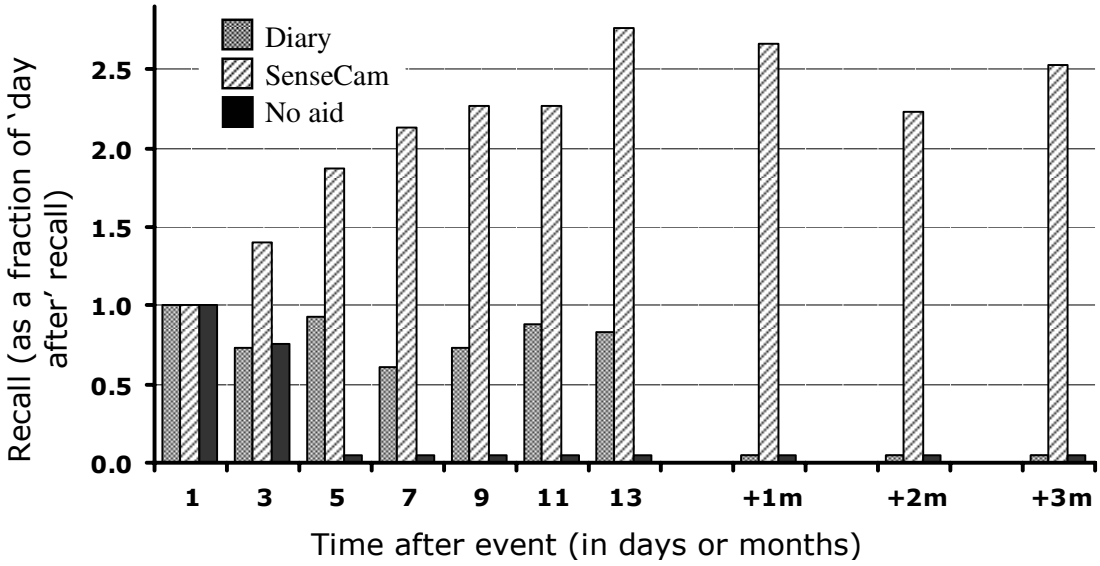


Figure 2.8: Effect of using visual life logging to a memory of patient with Limbic Encephalitis disorder.

in its memory. Hodges et al. [54], performed a study with a 63-year-old woman who was diagnosed with Limbic Encephalitis, a disorder of the brain's memory system. They compared three different scenarios: no aid, text diaries and visual diaries i.e. SenseCam. Fig. 2.8 shows the patient's relative recall of the details of the past events. The horizontal axis shows recall after a different number of days and the vertical access shows the relative recall. They showed that in average SenseCam helped the patient to remember nearly three times as many details as she could unaided.

2.6.3 Life Logging and Summarization

Another application of egocentric cameras is passive life logging. There are cameras that are specifically designed for life logging. It is required for these cameras to be energy efficient and easy to wear. Another limiting factor for these cameras is storage. These constraints impose many design challenges.

Other than the cameras' design challenges, processing the vast amount of images captured is very hard. Summarization is the key to managing large life logging image galleries. It is a way of reducing the size of the data in an abstract form e.g. sampled images [75], video subshots [82], grouped pictures [23], or high-level histograms [10]. The output of summarization can be representative, interesting or rare images/image regions. Another type of summarization is calculating different statistics from the visual or non-visual data such as activity histogram, concept statistics or location histogram.

Lee et al. proposed to summarize an egocentric video based on the appearance of important people and objects [75]. They defined a supervised learning problem to score regions based on their importance. They designed a two-step annotation process to ask annotators to find the important regions in their training video clips. The first step is asking the user to summarize a fast-forwarded video clip using a written sentence. Given the text description from the first step, a different user is asked to draw a segmentation boundary for objects and people mentioned in the text from the first step. This two-step annotation reduces the ambiguity of the task and makes it easier to explain to annotators. Typically, from a 3-5 hour video, they get 35 text descriptions and 700 object segmentations. Using these annotations, they taught a regressor to score regions based on the features that they extract from them.

The importance of regions is predicted from three different features: egocentric features, object features, and region features. Egocentric features are based on the region's distance to the center of the frame (similar to gaze), distance to hands and frequency (i.e. number of appearances of the region in the video). Object features are object-like appearance (e.g. regions perimeter and the difference between the object's texture and its surroundings), consistent motion different from nearby regions, and the likelihood of being a human face. Finally, the region features include region's size, centroid, bound-

ing box centroid and width and height. Using all of the mentioned features and their pairwise multiplications they create a feature vector for each region.

To summarize a video, first, it is segmented into multiple events based an agglomerative clustering on color histograms of video frames. Then given a compactness parameter for the summarization, each event is summarized with a set of selected frames based on the important regions that exist in those frames. To quantify their results, they asked people to compare their summarization results and a uniform sampling of the frames and showed that their method works better.

Aghazadeh et al. proposed a system that automatically finds novelties in life logging videos [1]. Their dataset consists of 31 videos of a subject walking from metro station to work every day. They leveraged the inherent repetitions in their dataset and matched video frames across different videos. For the matching, they used local features with PROSAC which is a faster method than RANSAC. With matching individual frames, they created alignment between these videos. Their idea for novelty detection is that whenever a novel situation happens, it cannot be found in other videos. They created their novelty detection based on this idea and compared their algorithm with ground truth provided by human input.

Alternatively, life-logging data can be summarized with some high-level information extracted from the images. Byrne et al. used a classifier to classify images into multiple categories to create a histogram of different concepts that exist in the data [10]. They used SenseCam cameras for their experiments and gathered a large dataset of a combined 137 days of 5 persons. They defined a set of 32 various concepts such as place concepts (e.g. buildings, indoor and office), objects (holding a cup, vehicle and holding a phone), activities (such as shopping, reading a newspaper, teaching and driving). They defined a fully supervised learning problem and used a bag of words approach

using Wiccest and Gabor feature extracted from images. Another common method to improve concept detection on life-logging data is smoothing the concept signal over time. The reason is that the same concepts tend to appear in multiple consecutive frames rather than a single frame. Therefore, smoothing of concept signals helps to improve the concept prediction accuracy. Using this method, Byrne *et al.* achieved 75% accuracy on their dataset.

2.6.4 Place Recognition and Navigation

Place recognition is another possible application of egocentric cameras. It can be used when either the positioning signals (GPS and WIFI) are low or when the exact position of the destination is not exactly known. Indoor environments are examples of situations where the GPS signal is not available and if WIFI signal is not available too, we need to use other sources of information to find the current location. Given a database of geo-tagged images, it is possible to search for an image to find its location. This is a well-developed research problem in Robotics which is called Visual Simultaneous Localization and Mapping (Visual SLAM). Similar approaches could be used for human mounted cameras with the main difference that human movements are usually much more unconstrained compared to robots. The search can be done just based on logos [110], street signs [90](see Fig. 2.9) or the whole image [35] [61]. Here we describe a few successful papers on egocentric place recognition.

Kang et al. designed a place recognition algorithm for indoor environments [61]. They formulated this problem as a large scale image search. For that, they performed local feature extraction with a term frequency, inverse document frequency approach (TF-IDF) approach. Their method consists of two steps. The first step is a normal TF-

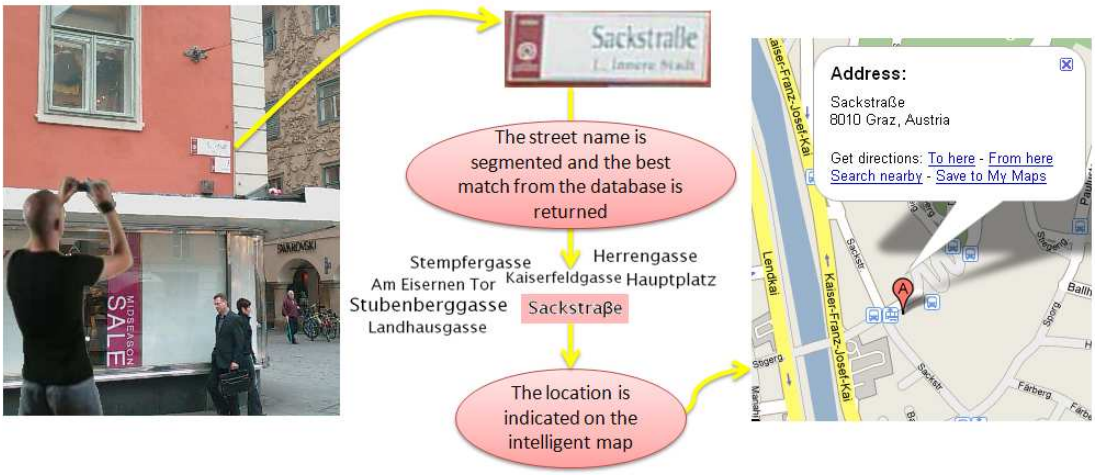


Figure 2.9: Overview of Naderi's place recognition system [90].

IDF search in which visual words are weighted inversely based on the number of their appearances. The output of the first stage is a set of candidate images. The second step, which is called Re-search, is essentially the same as the first step with the exception that inverse document frequencies are estimated based on the set found from the first stage. Using this approach, the gained about 15% improvement in average precision.

Flint et al. use a slightly different approach [35]. Their algorithm searches for the location up to a room by matching textons (i.e. vector quantized feature vectors) extracted from images. To decide on a location of an image they designed a probabilistic model in which correlations between textons in different parts of the image are used. The also applied the same method to classify whether the camera is looking downwards, straight or upwards. Additionally, they used this information to find out where to search for certain objects, which they call Active Search. For that, they found the correlations between object locations and the camera's orientation, which help in finding objects.

2.7 Conclusion and Future Works

In this report, we surveyed several recent papers in Egocentric Computer Vision. We first explained some of the basic tools and methods needed for the processing of egocentric data such as object recognition, hand detection, RFID, gaze, activity recognition as well as non-visual sensors such as EOG. We then listed the popular devices designed or used by industry or academia. Later we listed some of the public egocentric datasets. After that, we showed some applications that can be built on egocentric devices.

There are many ways to improve state of the art in Egocentric Computer Vision since it is in its early stages, especially in life logging which is one of the differentiations between Egocentric Computer Vision and the rest of Computer Vision. First of all, life-logging cannot be done with a non-egocentric camera. Secondly, there are certain characteristics that these videos have that can be exploited to create interesting applications.

One of these characteristic is that a human usually sees and interacts with a few instances of most of the object categories. For examples, the camera wearer sees an instance of each of the laptop, keyboard, monitor and computer mouse categories every day. There are exceptions to this too such cars and trees. A successful object detector should be able to leverage this fact and tailor its models for some of the categories to specific instances. Another interesting research problem is "how can we discover these objects?" and "how should we get annotations for those objects?", "how can we use existing annotated datasets such as ImageNet [18] for this task?" and "how do we use minimal user supervision to guide the process?".

Another characteristic of these videos is the daily repetitions. We need better methods to efficiently discover daily routines to be able to analyze life logging videos. Also,

finding events that are not a part of daily routines such as interesting moments and novelties will also be very beneficial.

Additionally, since battery and storage are limited to the wearable devices, programs that are running on these devices need to be energy efficient. Also, there is a fundamental question of how much data is needed for a certain task. It boils down to three factors: the video resolution, time sampling rate, and the video quality. We need to answer these questions to be able to save battery and storage space so as to increase the device's lifetime.

CHAPTER 3

DISCRIMINATIVE REGIONS: A SUBSTRATE FOR ANALYZING LIFE-LOGGING IMAGE SEQUENCES

3.1 Abstract

Life-logging devices are becoming ubiquitous, yet still, processing and extracting information from the vast amount of data that is being captured is a very challenging task. We propose a method to find discriminative regions which we define as regions that are salient, consistent, repetitive and discriminative. We explain our fast and novel algorithm to discover the discriminative regions and show different applications for discriminative regions such as summarization, classification and image search. Our experiments show that our algorithm can find discriminative regions and discriminative patches in a short time and extracts great results on our life-logging SenseCam dataset.

3.2 Introduction

We are entering the age of wearable computing. Wearable devices are becoming more powerful and ubiquitous. Wearable cameras such as Google's Project Glass, Narrative and SenseCam are adopted more and more by people. However, indexing the enormous amount of pictures that is being captured by these devices remains a challenge. Furthermore, these visual logs of people's everyday lives provide a rich source of data for information extraction, including a variety of different Computer Vision tasks. In this paper, we propose a method to find regions of interest in the life-logging image sequences and also showcase a few applications of this representation.

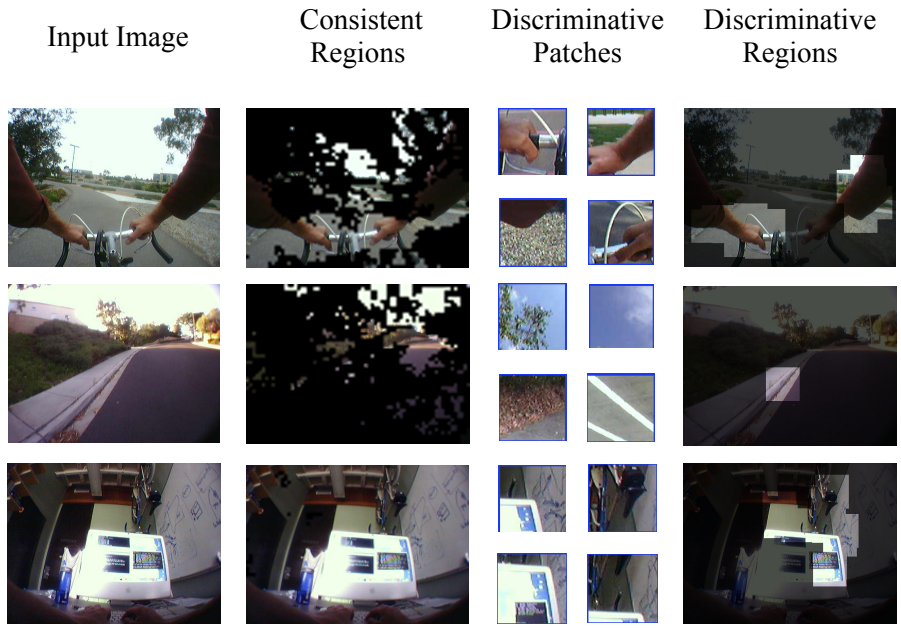


Figure 3.1: Discriminative Regions. First column is the input image. Second column shows consistent regions. Third column shows four discriminative patches discovered for set a of images from the same scene as the input image and Fourth column highlights the discriminative regions.

In this work we are using SenseCam, a chest-mounted wearable camera, that periodically takes pictures every 20-30 seconds. This results in about two thousand images per day. Thus, there is a need for algorithms to analyze and extract information from these image sequences to facilitate the search process. The process would ideally be unsupervised or supervised with minimal human input.

We propose an algorithm that highlights regions of interest in life-logging images. We define these regions to be *salient* i.e. conspicuous regions, *consistent* i.e. reliably appearing in a few consecutive frames, *repetitive* i.e. frequently appearing in the image set, and *discriminative* i.e. are specific to a particular scene.

Saliency and consistency constraints help focus on regions that are in the foreground. We design a novel robust method to find consistent regions based on a forward-backward

search in the feature space. Then those consistent regions are ranked based on their discriminative power and the number of their appearances. The motivation behind discriminative regions is that the parts that are visible everywhere are not very informative. Consider a scene where the user is working in his/her office. In this case, there are parts of the image that specifically belong to the office while others are shared with several other scenes. The idea is that the interesting regions are those that are specific to a scene and these regions should contain features that discriminate a given scene from others. Based on this definition, the office objects such as a monitor, keyboard, etc that are visible at the office become the discriminative regions for the office scene.

The organization of this paper is as follows. Sec. 3.3 reviews the related works. Sec. 3.4 and Sec. 3.5 explain our algorithm that finds regions of interest that satisfy the four requirements. Sec. 3.6 discusses how we find discriminative regions after discovering discriminative patches. Later in Sec. 3.7, we show a few applications we build on top of these discriminative regions. Sec. 5.5 discusses our experiments and results and we conclude in Sec. 3.9.

3.3 Background

Ego-centric, First Person or Wearable Computer Vision has recently emerged as an area of great interest to computer vision researchers. Recently there have been works on finding objects and their relations to activities [32], [33] and [104]. These methods usually require extensive amounts of annotation of object segments, bounding boxes or other labels for each image or video frame in the sequence.

Another relevant line of work is object discovery [112] [39] [19] [65] [124] [137]. Most of these works address category level object discovery or object category dis-

covery. These methods are designed for cases where multiple instances of each category are available in the dataset whereas in life-logging image sequences where there is usually one instance of each object category. The challenge in discovering objects in life-logging focuses more on finding objects in a variety of imaging conditions e.g. viewing angle, illumination, and occlusion rather handling the intra-class variation of objects such as different shapes or sizes of objects. Among these methods, Kang et al. work [62] is the most relevant paper. Their idea is to find groups of mutually consistent image segments. They first ran segmentation on all of the images of their dataset to get many small segments and designed an algorithm find co-occurring segments. Later, those segments are joined to form object segments. In this process, since their goal was instance-level object discovery, they also imposed appearance and geometry constraints in their optimization. They tested their methods on lab controlled image sets. The difference between their work and this paper is that we are working with real life-logging images and this introduces many challenges such as the size of the dataset, uncontrolled illumination, and occlusion. The other difference is that we focus more on finding candidate segment while their work is on how to connect these segments to form objects.

Our discriminative patch discovery method is similar to [21] and [123]. Singh et al. randomly subsample patches from a dataset of images with labels and apply an iterative discriminative learning method to find the patch clusters that demonstrate high discriminative power. Their iterative SVM learning is based on Ye et al. work on clustering [139]. We adapt the discriminative patch discovery idea and propose a simpler and faster algorithm to find discriminative patches.

3.4 Consistent Regions

As mentioned earlier, finding consistent regions is a crucial step in finding image regions containing objects of interest. We define consistent regions as parts that are visible across several consecutive frames in the image sequence. For example, in Fig. 3.3, the consistent regions are bike handle and hands while everything else lies on the background. The key factor is that the frame rate is variable between 20-30 as opposed to $\frac{1}{24}$ in SD videos. Due to the low frame rate, optical flow algorithms, object tracking, and background subtraction methods fail to find the consistent regions.

Life-logging scenes can be classified into two groups: stationary scenes and dynamic scenes. Stationary or low motion scenes are dominant in the life-logging images. These scenes are mostly sitting cases including working with a computer, watching TV, eating meals or sleeping and less occasional standing or other postures. Images of these scenes show very small change and large portions of these images are consistent. The other type of scenes is dynamic scenes in which usually the camera is translating in space. This includes biking, driving or walking. In the biking or driving scenes, there are parts of the image that are changing while some parts remain consistent. The consistent parts of the image may change their position in the next frames but they remain visible for at least a few frames. Walking or running are another types of dynamic scenes where the whole scene is significantly changing, do not have consistent regions.

There are several solutions for standard frame rate videos. Optical flow or background subtraction methods might solve this problem when the object movements are small. Fig. 3.3 shows a sample of three consecutive frames. In these frames, the bike's handlebar, some of the parts of bike's body and hands constitute the consistent regions while the road, trees, and sky compose the background. Conventional foreground-

background segmentation methods fail because of two reasons: (1) the appearance of those regions change between consecutive frames due to lighting changes, shadows, viewing angle and deformations. (2) the movement of these regions between these frames can be reasonably large e.g. the center of the bike handle may reside at the center of one frame and move to the far left in the next frame. Thus, we need a method that can handle more variations in the appearance and the movement of objects than the conventional methods do.

We propose a forward-backward search method to address these two issues which are based on searching small regions of the image in the images before or after it. To handle the location change, we search for each region in the next or previous frames in all of the possible locations. And we use HOG features to encode the structure of the region and cope with the variations in the appearance. Fig. 3.2 shows two consecutive frames from a biking scene. Consider a patch from the middle frame e.g. the center of the bike handle. We search for this patch in the previous frame to find its nearest neighbor. We do not constrain the search space to a local region and search throughout the previous frame for the best match. Our goal is to find another instance of the given patch in the previous frame if it is visible. One solution is to look at the distance between the given patch (A) and its nearest neighbor ($A^{'}$) in the feature space and accept the match as a correct match if the distance is lower than a threshold. But our experiments show that this approach is not very reliable, as the distances to the nearest neighbor for the wrong matches are sometimes lower than the correct matches. We propose not to rely on the distances in the feature space. Instead, we do another search for $A^{'}$ in the middle frame $A^{''}$. If the starting patch A is a consistent patch, then $A^{''}$ would be the same as or very close to A. And if the starting patch e.g. B is not a consistent patch, then its nearest neighbor, $B^{'}$, is a wrong match and $B^{''}$ falls at a different location.

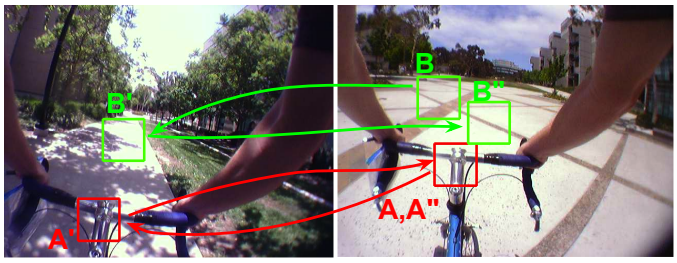


Figure 3.2: Consistent Patch Detection. Two patches from the right image are selected A and B. Their nearest neighbors in HOG feature space are labeled as $A^{'}$ and $B^{'}$ in the left image. Finally $A^{''}$ and $B^{''}$ are the nearest neighbors of $A^{'}$ and $B^{'}$ respectively.

To implement the proposed method, first, we extract dense HOG features from a grid of points from middle and left frames and create a kd-tree on the extracted features for each frame. Using the kd-trees, two searches are done for each patch in the given image. If the resulting patch is spatially close to the given patch, they the given patch is considered a consistent patch. The same procedure is done on the middle and right frames, and the final consistency map is the intersection of the two maps. Fig. 3.3 shows threes consistency maps where the middle one is the intersection of the left and right maps. Finally, the last image shows the middle image with the overlaid consistency mask.

3.5 Discriminative Regions

After finding the salient and consistent regions, our goal is to group these regions to form clusters. Some of these clusters may represent meaningful regions and some clusters do not, even with very low intracluster error. We define meaningful clusters as those who demonstrate discriminative power. For the discrimination, we use two sets (labels). Since we use image labels, this work can be considered as a weakly supervised method.

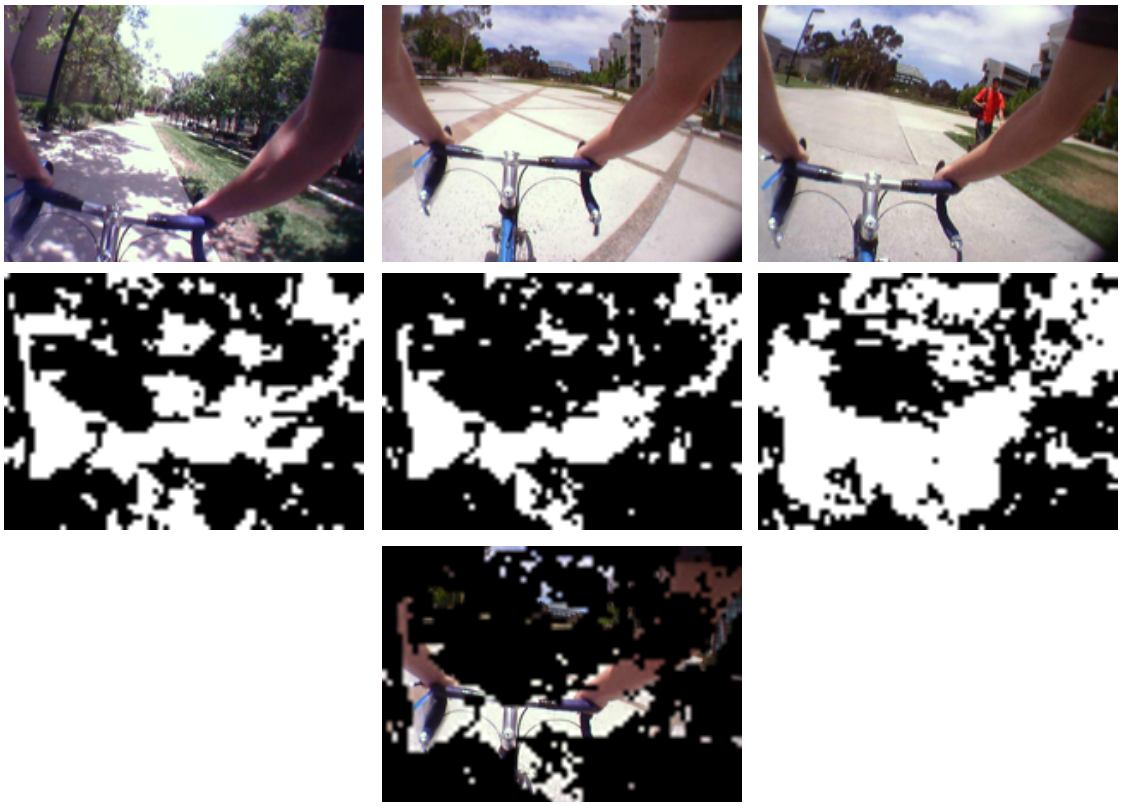
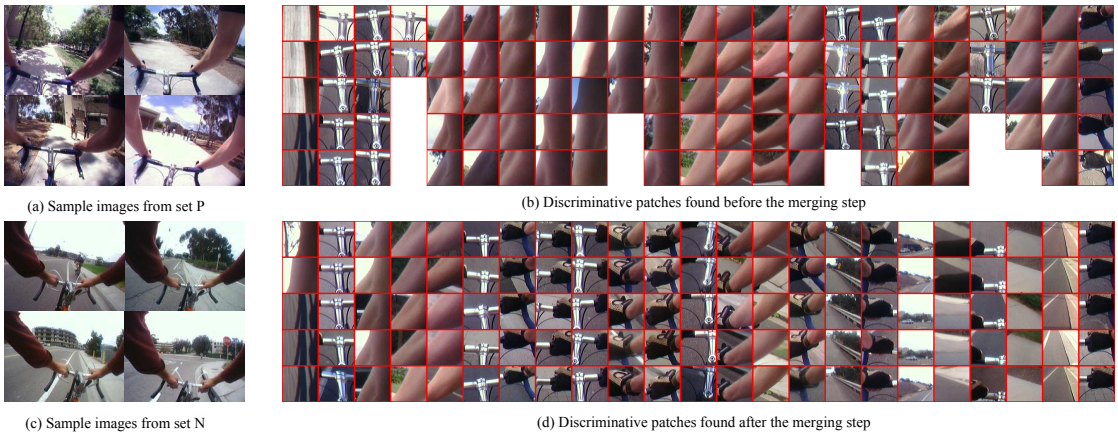


Figure 3.3: Consistent Region Detection Sample. The top row show three consecutive images from a biking scene. The second row shows the consistency maps. The right binary map is generated using first two images and the left map is generated using the last two images. The middle map is the intersection of the two maps which is overlaid on top of the original image in the third row.

Later we discuss ways to automatically extract the information we need for these labels.

We assume that we have two image sets: $\mathbb{P}$ and $\mathbb{N}$. $\mathbb{P}$ contains a group of images from a particular scene. The scenes can be dynamic or static, and they usually correspond to human activities such as biking, walking, driving, working in the office and watching TV. $\mathbb{N}$ is the universal or negative set and contains everything but $\mathbb{P}$. For example $\mathbb{P}$ may represents images of a biking scene while $\mathbb{N}$ is the set of all other images. Later in the Sec. 5.5, we discuss different combinations of $\mathbb{P}$ and $\mathbb{N}$.



(a) Sample images from set P

(b) Discriminative patches found before the merging step

(c) Sample images from set N

(d) Discriminative patches found after the merging step

Figure 3.4: Discriminative Patch Discovery. (a) and (c) show examples of $\mathbb{P}$ and $\mathbb{N}$ respectively. (b) shows the discriminative patches that are discovered by our algorithm. Each column represents a discriminative patch and the patches in each column are the nearest neighbors of the patch in the first row. Finally, (d) shows the discriminative patches after the merging step.

$$\mathbb{U} = \mathbb{P} \cup \mathbb{N} \subset \mathbb{R}^d \tag{3.1}$$

More specifically $\mathbb{P}$ and $\mathbb{N}$ are the sets of extracted features from patches randomly sampled from the consistent and salient regions. We used histogram of oriented gradients (HOG) [16] and color ($a^\star$ and $b^\star$ channels of $La^\star b^\star$ color space) to represent each patch. Furthermore, all of the selected patches from an image share the same labels as their image source. There is a one-to-one mapping between labels and sets. All of the patches in $\mathbb{P}$ have $+1$ label while patches in $\mathbb{N}$ have -1 label.

We propose to solve the discriminative patch discovery by looking at nearest neighbors of each feature point. $N_k(x)$ is defined as set of k nearest neighbors to x based on the distance function d. If we look at the distribution of labels in $N_k(x)$, we can find patches that most likely appear in either $\mathbb{P}$ or $\mathbb{N}$ and patches and appear in both sets.

$$N_k(x) = \{x' | x' \in \mathbb{U}, \forall y \in \mathbb{U} - N_k(x) : d(x, x') < d(x, y)\}$$
$$and |N_k(x)| = k \tag{3.2}$$

Specifically, we measure the label proportion in $N_k(x)$ by counting the number of element in each set (i.e. $\mathbb{P}$ or $\mathbb{N}$) and dividing the two numbers. The larger the value of division, the more discriminative the patch. The idea of iterative discriminative learning [21], [123] is to first run unsupervised clustering such as k-means and then at each iteration train an SVM classifier to separate each cluster from the rest and take the top matches of each classifier each cluster as positive in the next iteration to train another set of classifiers. Following their method, we implemented the iterative SVM learning but we did not see improvements over our nearest neighbor based discriminative patch discovery. There are two main reasons. The first reason is in the nature of our data. Since we are using life-logging images, there is an inherent repetition of objects in the image sequence that helps to achieve good clusters with our method. The second reason is that our results are much cleaner than k-means clustering which makes it hard for the iterations to improve the quality.

$$D(x) = \frac{|N_k(x) \cap \mathbb{P}|}{|N_k(x) \cap \mathbb{N}|} \tag{3.3}$$

After calculation of the discriminativeness value $D(x)$ for all the patches in the $\mathbb{P}$, we sort patches based on their discriminativeness value in the decreasing order and pick those with highest values. In our experiments, we use $D(x) = 4$ as a cut-off value to select the discriminative patches. $D(x) >= 4$ is equivalent to have more than or equal to 80% of the nearest neighbor points in $\mathbb{P}$.

$$\mathbb{D} = \{x | D(x) > threshold\} \tag{3.4}$$

Consider a patch that has a high discriminativeness value. It is expected for its nearest neighbors to have high discriminativeness values as well. The reason is that in that case, the discriminative patch resides in a part of the feature space that is mainly filled with patches from $\mathbb{P}$. Because of this, some of the elements of $\mathbb{D}$ become very similar. Thus, we need to remove and merge some of the clusters. The last step is to merge the clusters that have a significant amount of overlap. If two clusters are to be merged, we remove the cluster center that has less discriminativeness value from $\mathbb{D}$ and add its cluster members to the nearest neighbors of the cluster with higher discriminativeness value.

3.6 Spatial Relations Among Patches

Depending on the definition of objects and their sizes, the discriminative patches that we find may represent whole objects or object parts. When the patches are smaller than an object, we need to link them together to build objects or discriminative regions. For that, we need to find the spatial relationship between discriminative patches.

We experimented with different methods to find spatial relationships among patches. The best method turned out to be frame selection. In this method, we find images that contain a high number of discriminative patches and deduce about the patches geometry based on their placements in the candidate frames. Specifically, we rank the frames based on the number of appearances on the discriminative patches in them. This is a very fast process since we only need to go through discriminative patches and their nearest neighbors and all of those patches have pointers to their original image frames. Fig. 3.5 shows four frames that very highly ranked for the biking scene.

Figure 3.5: Frame selection to visualize spatial relationship among discriminative patches. Note that highlighted regions are discriminative patches found in the top ranked frames.

3.7 Applications

There are many uses for the discriminative regions we present in this paper. These regions depict information about the scene and the activity that is being performed. These regions are of importance in many applications such as classification of images, summarization, and content-based image search.

Summarization. Since discriminative regions contain object and foreground information, they carry high-level information about the activity that is being done from the image sequence and can be used for summarization of high-level concepts. The idea is to select a minimal set of frames and cover all the discriminative patches. In this way, most of the activities in the image sequences will be summarized with a small number of images.

Classification and Statistics. Discriminative Regions, by definition, are suitable for classification. Our algorithm can highlight regions that have discriminative power and can be used for classification. One example classifier would be a linear classifier on the histogram of discriminative patches found in images.

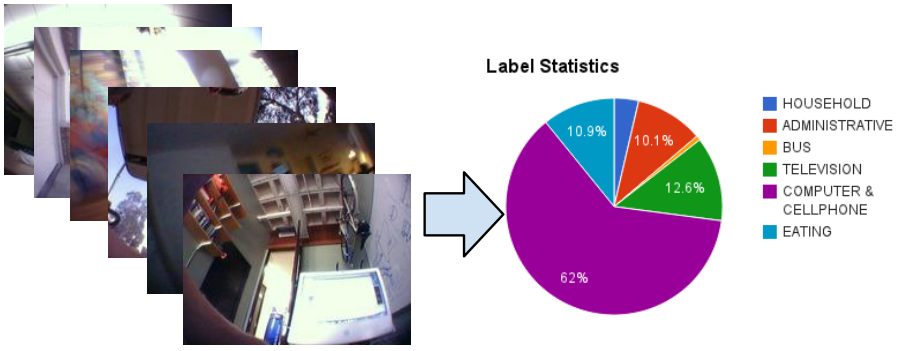


Figure 3.6: Example images and a sample of histogram of daily activities.

The result of classification can be turned into a histogram of activities or scenes e.g. Fig. 3.6. This will give an excellent overview of a given day and can be accompanied by an analysis of physical activities and recommendations about personal health.

Labeling and Search. The idea of labeling and search is similar to face labeling in Google Picasa and iPhoto. First, these programs detect faces in image galleries and cluster them into face clusters. They then ask the user to label representative faces from each cluster and using those label they label other faces in the image gallery.

In life-logging image sequences, we are not only looking for faces but anything else that are frequently appearing in the image set. After finding discriminative patches, we ask the user to label the representative patch from each cluster. Since the discriminative patches we find might be a part of an object, we show the whole image and highlight the discriminative patch and ask the user to label the enclosing object. The reason is that a cropped patch might not be informative enough to be perceived by the user but given the context, it becomes easy to label. In this process, the user can choose one of the previously chosen labels or add a new label. In this way, we control the number of labels and discourage the user from adding an excessive number of labels. Then, our algorithm propagates the labels to all the instances in the clusters. Fig. 3.7 demonstrates

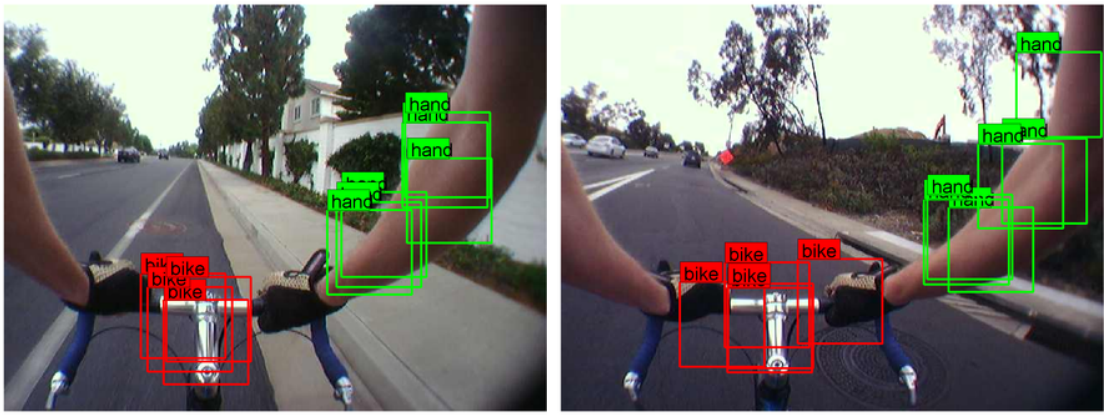


Figure 3.7: Example images with the two propagated labels: bike and hand. We manually labeled top cluster centers and ran our algorithm to propagate the labels to all of the members of those clusters. Green boxes are hand labels and red boxes are bike labels.

a few images with the propagated labels. Once we establish labels for images, we can use them for search and ranking. For Fig. 3.7, we ran our software and labeled the top 100 discriminative patches and our algorithm propagated the labels to all the candidate frames.

3.8 Experiments and Results

3.8.1 Data Acquisition and Annotation

The participants were adult cyclists recruited through a university-based cycle-to-work network. Eligible participants were aged 18-70 years, were university employees, routinely bicycled for transportation. Each participant wore the SenseCam during waking hours for 35 days. They were instructed to perform their normal daily living activities, and turn off SenseCam in private time (e.g., bathroom). Excluding night time or pri-

vate pictures that were removed before doing our experiments, our dataset contains over 360,000 images.

All of the images in our dataset, are labeled with position labels (i.e. Sitting, Standing Still, Walking/Running, Biking), and activity labels (such as Household Activity, Administrative Activity, Television, Other Screen Use and Eating). To the best of our knowledge, this becomes the largest SenseCam dataset with reliable labels. Our dataset is available on our lab's website[1].

3.8.2 Implementation

The first step is the consistency detection. For that, we randomly select 400 images from $\mathbb{P}$ and $\mathbb{N}$. Then, first, we extract dense HOG from a grid of 70×50 points from all of the selected images. We chose to extract $8 \times 8 \times 31$ bin HOG from patches cropped around each point on the grid using HOG implementation of Girshick et al. [42]. Note that, throughout this paper; we use a fixed patch size of 100×100 pixels. This size was selected to represent a cover a good size for object parts while the images have a resolution of 640×480 pixels.

For consistency detection of each image, we need to extract HOG features from three images. After that, we search for all of the feature points in the given frame and frames before and after that and vice versa. Thus, we need to perform a total of four searches that are implemented using [25]. Assuming we have run this process for a given image, for the next image, we only need to run the feature extraction on the new image and do two searches corresponding to the new image. Using this technique, we achieved the speed of about 5 seconds per image on a MacBook Pro with 2.9 GHz CPU. We also do

[1] http://vision.ucsd.edu/%7emohammad/sensecam_dataset

this process for all images but with two frame distance i.e. frames $i-2$, i and $i+2$. The consistency maps are later combined to be used in the patch sampling process.

Having the consistency maps computed, we extract 50 patches from each image that is selected for each study. The extracted patches have to satisfy two requirements: saliency and consistency. We use a low-level saliency definition which is defined as a minimum standard deviation as well as a minimum on the length of the vectorized HOG feature. For meeting the consistency requirement, we randomly sample based on the binary consistency maps. The result of this step is about 20000 HOG-color feature points.

Then we compute the distance between all pairs using [25] and pick the top 50 nearest neighbors for each point. These nearest neighbor are then used for the ranking of discriminative patches. The whole process takes about 5 minutes assuming that the consistency maps are pre-computed.

3.8.3 Experiments

The first experiment is to show the effect of consistency maps. In this experiment, $\mathbb{P}$ is a set of biking image of a particular person where $\mathbb{N}$ is a set of images from an office scene. Fig. 3.8 show the discriminative patches that are discovered in this experiment. The top row is the discriminative patch discovery without using consistency detection. For second and third rows we used three and five images for consistency detection. As it suggests, using consistency maps helps to focus more on the foreground object rather than background regions such as sky, trees or road. It also shows that using more frames for consistency detection results in better foreground object clusters. Additionally, since the patch sampling is more restricted and the number of discriminative patches becomes

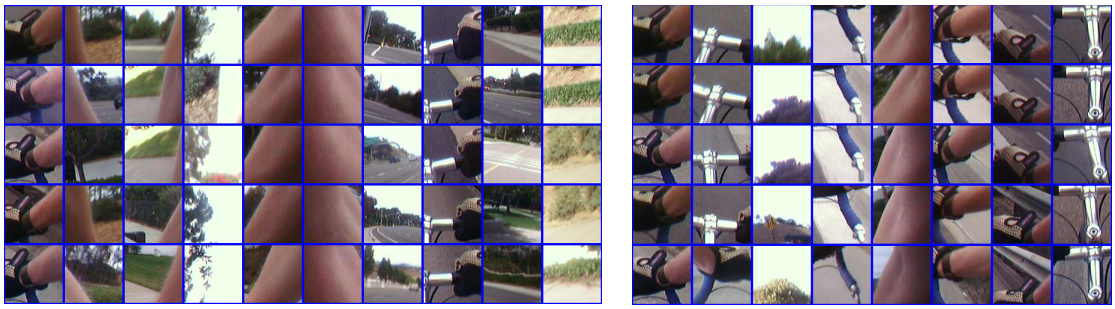


Figure 3.8: Effect of using consistency detection. The left part shows the discriminative patches without using consistency detection and the right part demonstrates the effect of consistency detection with five images. Using five images reduces the resulting discriminative patches that are mostly foreground patches.

less.

The second experiment is studying the effect of type of scene. For this experiment, we have used five different labels: Car, Biking, Watching TV, Walking/Running and Sitting. The image where chosen the image sets of two different persons. In each case $\mathbb{P}$ is defined to be the set of images having each of the five labels from one person's data and $\mathbb{N}$ is the images from the same label but from the other person. Fig. 3.9 shows the discovered discriminative patches and discriminative regions from the top image selected by our frame selection process.

The Car and Biking scenes are similar in nature. There are some parts are visible in most of the images and some parts that are in the background. Fig. 3.9 shows that our algorithm can detect bike's parts, hands as well as parts of car's dashboard. The Watching TV and Sitting scenes are stationary scenes and our algorithm discovered monitor, TV, TV stand and well as other objects in the two rooms. Walking/Running is a completely dynamic scene and the set of discriminative patches are very small compared to other labels and our algorithm focused at a fence, some parts of sky and lamps that

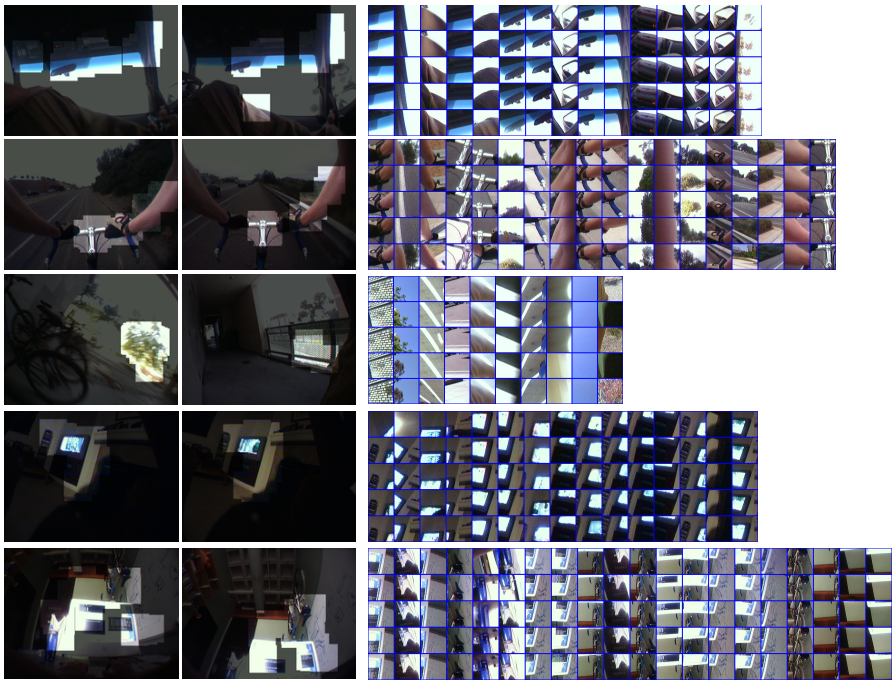


Figure 3.9: Discriminative patches and discriminative regions highlights for different scenes. The scenes are Car, Biking, Watching TV, Walking/Running and Sitting from top to bottom respectively.

seems to be specific for one person.

3.9 Conclusion

In recent years, there has been a surge of interest in wearable cameras in the industry. However, there is a need for efficient and accurate image analysis techniques for processing life-logging images. In this paper, we presented a method to extract discriminative regions with minimal supervision. We also hinted at how these discriminative regions can serve as a substrate for life-logging applications. Finally, we showed a few applications we can build on top of discriminative regions such as summarization and improved search and object part detection.

CHAPTER 4

EXPERIMENTS ON A RGB-D WEARABLE VISION SYSTEM FOR EGOCENTRIC ACTIVITY RECOGNITION

4.1 Abstract

This work describes and explores novel steps towards activity recognition from an egocentric point of view. Activity recognition is a broadly studied topic in computer vision, but the unique characteristics of wearable vision systems present new challenges and opportunities. We evaluate a new challenging publicly available dataset which includes trajectories of different users across two different indoor environments performing a set of more than 20 different activities. The visual features studied include compact and global image descriptors, such as GIST and a novel skin segmentation based histogram signature, and state-of-the-art image representation for recognition, including Bag of SIFT words and Convolutional Neural Networks (CNN)-based features. Our experiments show that simple and compact features provide reasonable accuracy in extracting basic activity information (in our case, manipulation vs. non-manipulation). However, for fine-grained categories CNN-based features result in the most promising approach. Future steps include the integration of temporal consistency into the pipeline and depth information into the CNN-based descriptor.

4.2 Introduction

Thanks to the advances in consumer electronics, digital cameras are ubiquitous sensors whose presence is always growing and offer more and more solutions to real-life problems. Besides, the miniaturization of the camera optics and electronics has facilitated

the construction of all types of wearable visual sensors. The ever-growing computing capabilities, either locally or in the cloud, are pushing even further the potential for this technology. There is a large variety of applications that one can think by understanding the information captured by cameras worn by a person, mounted or attached to a human body. From early prototypes often focused on life logging, e.g. [55], to more interactive devices such as Google Glass[1], advancements in the design of these wearable cameras/computers is pushing the boundaries of wearable vision systems applications. In this paper, we want to explore the opportunities of vision and depth (RGB-d) sensors on this field, in particular for ego-activity recognition.

It is often a real challenge to distinguish between fine-grained activity labels if we only use still frames (e.g., opening/closing a door). Sequential and temporal information is the key to distinguishing some of those cases, but still in many activities still images can provide a good idea of what may be happening. This work is focused initially on classification of still images (i.e., the descriptors computed separately for each frame), but

The main goals and contributions described in this work are the following:

- Evaluating a prototype with an RGB-d helmet mounted camera which was used to acquire a set of multiple users performing indoor activities in a similar manner/environment. We propose a hierarchy of labels to facilitate some contextual information before running a fine grain activity classification.
- Analyzing the performance of new image representations for activity recognition, using only still frames from the given sequences. On the one hand, this work evaluates a set of proposed compact global descriptors built after the proposed skin segmentation steps. On the other hand, we evaluate the performance of the

[1] `http://glass.google.com/`

recently spread Convolutional Neural Network based image representation for activity recognition.

4.3 Related Work

Wearable cameras are getting more and more common and therefore there is a recently increased interest in the computer vision applications that can be developed or adapted to these specific settings. We find most of the earlier works to be focused on visual logging, *life logging*, applications [98]. So we find a computer vision task that turns a necessity for logging purposes is the ability of automatically summarizing those videos, as it was studied in [76].

Some other works focus on shorter-term wearable camera applications, such as user interaction. One of the earlier works in this topic presents an approach to analyzing the hand location and gestures to understand the user interaction [86]. Also focused on the user interaction, we find a more recent proposal to monitor the user workspace from an RGB-d wearable camera [17] or an approach to track people moving around the user from a wearable RGB-d vision system [93]. Our work explores as well how to take advantage of RGB-d wearable sensors, but in the context of activity recognition.

In the last years we can find interesting proposals related to this goal of activity recognition from an ego-centric point of view, for example, to recognize the interactions that the user is suffering [113] or on how to take advantage of the point of view of the camera to estimate the gaze of the user, as the information about where the user is looking is a strong hint for activity recognition [79].

Action recognition is an important topic in computer vision, not necessarily from an

egocentric perspective, so many of the lessons learned in general settings are of interest for our work. In particular, works on skin segmentation [133, 102] are an important element in our work. Skin pixels will contain very important information if we analyze the interactions of a person with the environment, but they will not be displayed or arranged in the same manner in the image if they are captured from a wearable or non-wearable camera. Therefore, as we will see in later sections, we build our image description starting from the pixel-wise skin segmentation rather than from hand detection libraries, typically designed for another point of view (frontal).

Another group of works that inspire partially our approach are scene understanding approaches. Before analyzing in detail the content of a scene, an initial step is usually to discover the type of scene. It has been shown in many applications, that global image information can be used to provide context information into the finer grain recognition system [130]. These hierarchical recognition ideas have also been shown to be useful in the context of activity recognition with spatio-temporal features [126].

In this work we evaluate compact image description for an initial activity understanding, and explore how state of the art features on recognition problems behave for activity recognition tasks. Convolutional Neural Networks (CNN) proved their performance on various computer vision dataset such as the largest labeled object recognition dataset, Image Net[18]. In 2012 Krizhevsky *et al.* achieved the best results on ILSVRC2012 [70] which followed by many works in this domain. Donahue *et al.* implemented a CNN in their system (DECAF)[26]. Recently, Razavian *et al.* did experiments with CNN features on different Vision datasets such as PASCAL VOC, UCSD, CUB200 and Oxford Buildings and showed the CNN features based on a model that is trained for ILSVRC either lead to state-of-the-art or are competitive with other methods[107].

4.4 Hierarchical Activity Recognition from a Head-mounted camera

As previously mentioned, many recognition systems that need to handle large amounts of data work in a hierarchical way. First, the system prunes the classification options using global information, to either find a set of possible candidates or provide some context information; then, a more detailed analysis is performed to determine which of the potential candidates is the best fit.

In this work, we propose how a similar hierarchical process could be useful for the task of activity recognition from a wearable vision system. The following proposal assumes a computer vision system with similar properties to the one we are testing in this work: a head-mounted camera, pointing a bit down, to facilitate keeping the user hands within the field of view and therefore allowing the analysis of the actions performed. We can see the described configuration in Fig. 4.1.

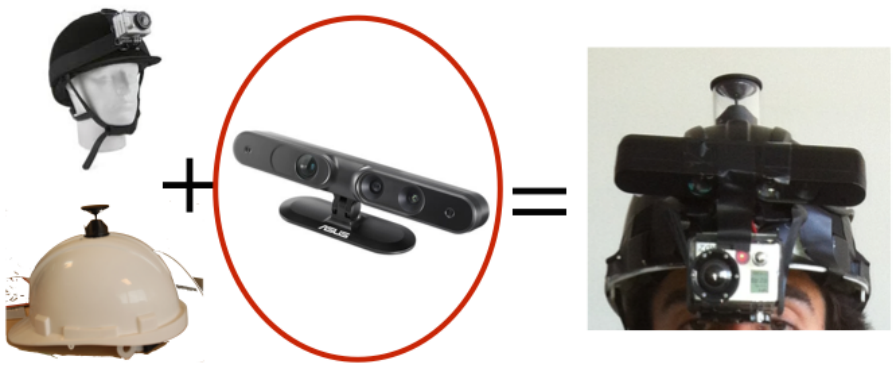
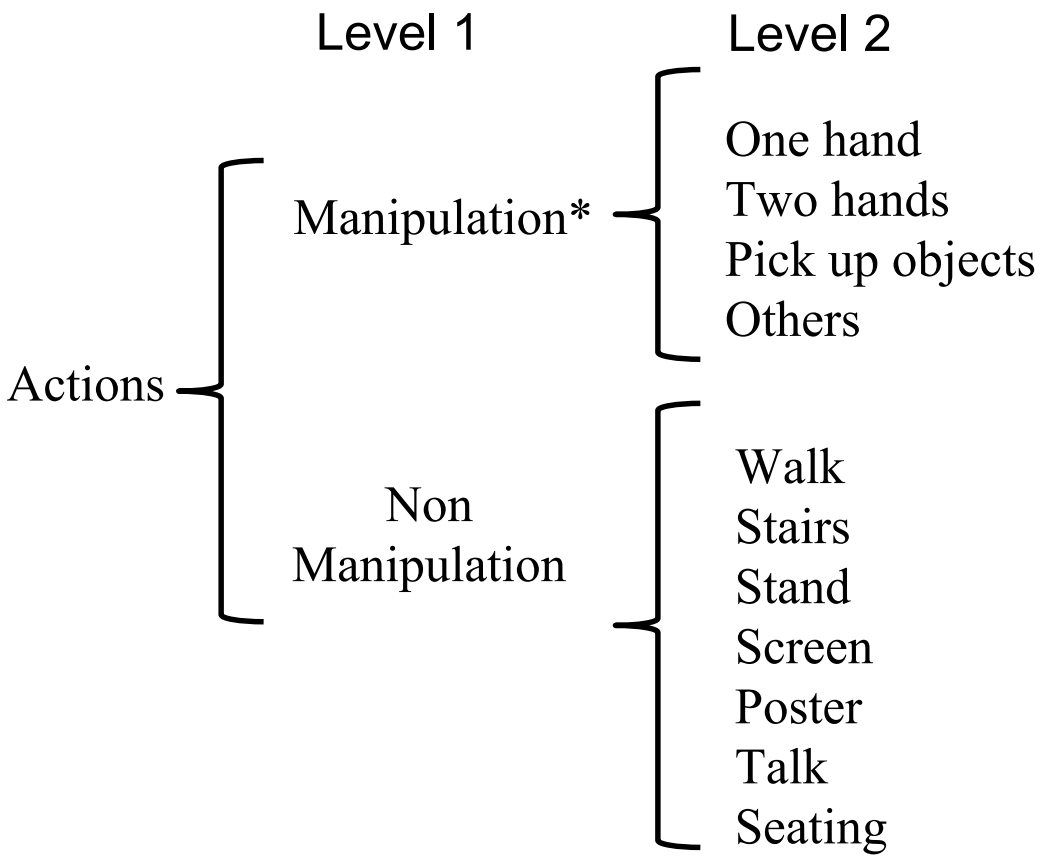


Figure 4.1: Wearable vision system evaluated in this work. We use an RGB-d camera mounted in a helmet together with other vision sensors (not used in this work).

Figure 4.2 summarizes the labels we are considering at different levels of granularity. At level-1, we aim a binary classification into manipulation and non-manipulation



* Level 2 - Fine grained **manipulation** labels:

Two-hands	includes all activities where the user uses/shows both hands while manipulating something.
One-hand	includes all activities where the user uses/shows just one hand while manipulating something.
Pick-up	includes all activities where the user picks up or drops an object.
Others	all activities that imply manipulation but somehow do not fit any of the above.

Figure 4.2: Hierarchy of Action Labels used in this work.

actions, since from an ego-centric perspective is easy and convenient to analyze what the person is manipulating, especially if the wearable vision system is already designed for this task (e.g., the system we are testing, a helmet-mounted camera pointing a bit

downwards). Level-2 and level-3 model progressively fine-grained categories within each group.

Since the non-manipulation labels are continuous along all the trajectory, in some frames we find that several labels could occur simultaneously. In those cases, we consider for each frame only the dominant label (e.g., if the user is seating and reading from the computer screen, we assign the *Screen* label to that frame).

Labels in the dataset used include an additional level-3 that details further the manipulation activities but is not used in this work due to the low number of occurrences per action (less than 1 per sequence) of many of them within the labeled data. As shown in the following sections, on level-2 the performance is not good, and both more sophisticated descriptors and examples from each activity are needed to provide the classification system with the necessary training information and discriminative power. Level-3 in the dataset includes the following sub-divisions: Two-Hands contains Typing in a keyboard, Using the mouse, Reading a paper, Reading a book, Handwriting on paper; One-Hand contains Hand-shaking, Writing on board, Open-close door, Open-close window, Open-close fridge, Open-close microwave, Open-close closet; Pickup objects label contains Using vending machines, Using the phone, Drinking, Eating, Pickup, and Drop object.

4.5 RGB-d image segmentation and description

As previously described, this work is focused on how much of the activities that are happening can be identified from still frames. This section describes the different types of image descriptors proposed or studied in this work for the activity recognition experiments.

4.5.1 Global image descriptors for scene understanding.

The first type of descriptors considered is typical global image descriptors, which give a compact image representation frequently used for scene categorization and place recognition, such as GIST [96], color histograms or global image statistics such as color invariant moments.

4.5.2 Skin segmentation based features.

Since our main focus is the egocentric activity recognition, we use our framework constraints (we are working with a head mounted camera pointing down) to design specific features that roughly encode the distribution and location of the arms and hands in the image. We start from a standard color based image pre-processing to segment out the skin pixels [133] in the image. In particular, we apply the following filter, (4.1), to the RGB values of each pixel, and if a pixel matches all the conditions it is considered as a skin pixel:

$$(R > 95)\&(G > 40)\&(B > 20)\&((MAX(R,G,B) - MIN(R,G,B)) > 15)\&(|R-G| > 15)\&(R > G)\&(R > B). \tag{4.1}$$

Additionally to this color segmentation, since we are using a depth sensor, we can filter the arms and hand pixels not only by the skin color but also using the depth. We establish a threshold for the maximum distance where the hands and arms appear (measured out of 5 different users, we set this threshold to one meter). As we can see in the Fig. 4.3, this helps to get a better skin segmentation. The inclusion of depth information allows us to be less strict with the range of colors accepted, but indeed

the skin color segmentation is still very dependent on the users participating in our experiment. Therefore a user-based skin color calibration will be needed in a more general setting.

We have also considered additional steps: plane segmentation and superpixel segmentation. The step to remove the dominant plane found in the image sounds promising, but in our initial experiments, the computational cost increase was too high for the improvements obtained, so it is not used in the experiments in this work. However, an additional superpixel segmentation step helps us avoid discontinuities in the skin segmentation: we run a fast superpixel segmentation step, using SEEDS superpixels [132] and assign skin or not skin to each superpixel depending on the average RGB color of the superpixel components.

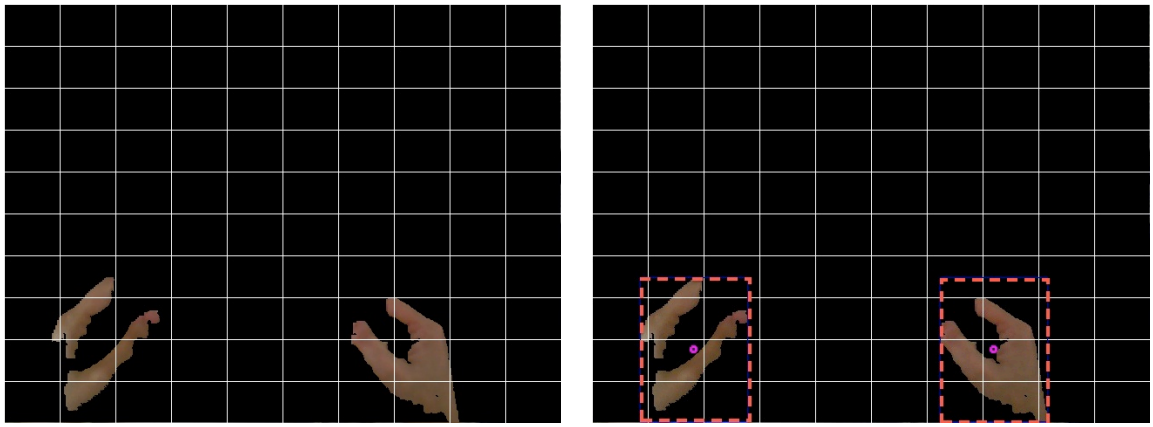
(a) (b)

Figure 4.3: Skin segmentation. (a) Using only color filtering (b) Using color and depth filtering. The white pixels (those that are NOT within the red dashed rectangle) were accepted by the color filer but rejected by the depth filter.

Given the final skin pixel segmentation, we have designed a variety of descriptors that build on top of it, from those, the most promising ones are:

Skin histogram ($SKIN\_HIST$)**:** we divide the image in a 10x10 grid, as shown in Fig. 4.4(a), and build a histogram that represents the ratio of skin pixels contained in

each cell.



(a) (b)

Figure 4.4: Skin segmentation based descriptors. (a) Skin histogram descriptor obtained from a 10x10 grid on the skin-segmented image. (b) Bounding boxes obtained in a sample image are represented by red dashed rectangles.

Arms-Hands Bounding Box (*BB*): we detect the two largest connected components of skin pixels, which are over a minimum size, and compute the bounding box around them as shown in Fig. 4.4(b). We describe each bounding box with the following two descriptors:

- We ran a PCA on all the skin pixels within each bounding box to obtain the eigenvectors, which will be used as a measure of the *shape and orientation* of the set of skin pixels in that bounding box.
- We compute the ratio of skin pixels in the bounding box, to measure the *density* of that bounding box content.

4.5.3 Convolutional Neural Network (CNN) Based Image Representation

Finally, we are exploring the performance of Convolutional Neural Networks based image representation for activity recognition. Convolutional Neural Networks (CNN) proved their performance on various Computer Vision benchmarks[107]. Razavian et al. showed that CNNs are very effective in representing images for a variety of object and scene recognition tasks. In this work, we use the implementation of [26] (DECAF). We use a pre-trained model that is trained to classify 1000 object categories of the Image Net 2012 challenge. We run feed-forward pass for all of the video frames of our dataset and use the seventh layer's output to represent the input frame.

After representing images with a 4096-dimensional feature vector, we trained a linear SVM[29] to classify our labels such as manipulation/non-manipulation or different hand manipulation labels. Even though the network is trained to classify objects, the features seem to be general enough to give the best results on still image activity recognition as well.

4.6 Experiments

4.6.1 Dataset

The data we use in this work is part of the *Wearable Computer Vision Systems dataset* recently acquired with the purpose of comparing different wearable cameras for different wearable vision system applications. This dataset is available online[2]. We use the 5

[2]Web link is removed due to double-blind policy

labeled sequences available using an RGB-d camera. These were acquired by 4 different users in 2 different scenarios, where the users performed a set of actions as they were told but without a specified order and at their own pace. These sequences were manually labeled with different granularity level for activity labels as well as with location labels (not used in this work) and they present a challenging classification problem with large intra-class variations (due to multiple users and scenarios).

4.6.2 Experimental setup

We have run a leave-one-out cross-validation for our experiments. We trained classifiers based on the data from 4 sequences and tested it on the other sequence data. The final performance is reported as the average results across different tests. We use accuracy and confusion matrices to evaluate the different image representation and classification methods proposed in this work as baselines.

Classifiers and features used

The following combinations of descriptors (detailed in Section 4.5) are considered in our experiments:

- GIST descriptor. Typically used for scene categorization, we use the implementation available from [83].
- Skin histogram ($SKIN\_HIST$). It is obtained from skin segmentation obtained from color, depth, and superpixel filters.
- Bounding Box (BB). We compute this descriptor on the bounding boxes around the two largest skin connected components found (if they exist).

- CNN: We run feed forward path for each frame in the video and take the output of the seventh layer as the representation.
- CNN-MULTIWINDOW: This is the same as CNN representation with the difference that the CNN feature extraction was run on 5 windows of the images: the four corners and the center part. The representations are concatenated to make the full representation.
- SIFT - BOW: We extracted Dense SIFT[80] features and used 1000 cluster center to learn 1000 visual words. Then with histogram encoding, input images are represented by a 1000 dimensional feature vector.

4.6.3 Performance evaluation

We present the experimental validation of our study organized in two sets of experiments. The first set analyzes the performance of compact and global descriptors and the differences observed when using different classifiers. The second set explores the performance of additional more sophisticated image representations and compares all the baselines proposed.

Performance of global image descriptors

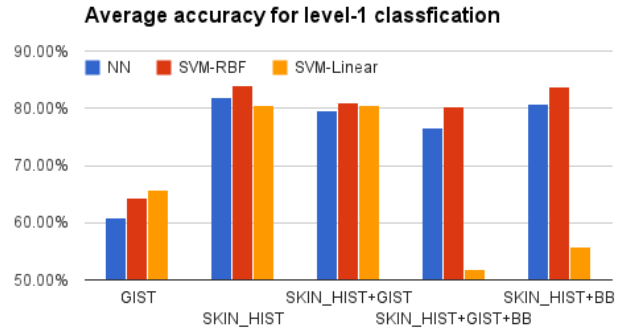
Our first set of experiments analyzes the different configuration of the global and compact image descriptors described in Sections 4.5.1 and 4.5.2. The goal is to evaluate which classification framework would be more suitable for these descriptors and the discriminative power of each of them for the activity recognition. Since these descriptors are compact and efficient to compute (just one descriptor per image), each combination

has been used in three different classification frameworks: nearest neighbor (*NN*), linear SVM (*SVM-L*) and SVM with RBF kernel (*SVM-RBF*).

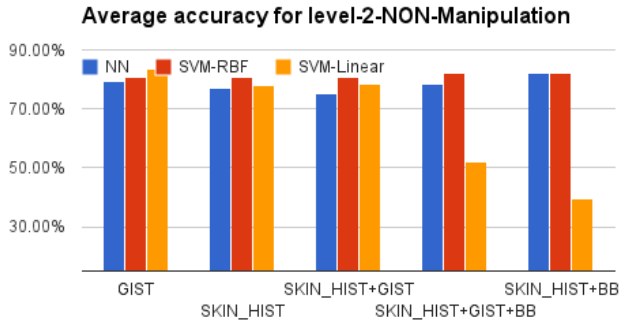
Single step classification. As a motivation example for the hierarchical process we propose, we ran a baseline experiment that consisted on training a single classifier for all the eleven level-2 classes at once, with different combinations of descriptors and classifiers. The best results obtained with the different combinations run were a raw accuracy (total number of correctly labeled tests divided by total amount of tests) around 35%. However, if we compute the accuracy normalized per class, it drops to around 15% (slightly better than chance). This means that the discriminative power of those features and the available amount of training data for each of those classes is not enough to directly distinguish among all of them. The classifier ended up assigning almost every test to the dominant test class. Note that the training was done as balanced as possible, but still, too many classes did not have enough occurrences in the dataset.

Classification in two steps. If we run the two-step classification proposed, we obtain more promising results. Figure 4.5 shows the classification performance achieved in the two considered levels of the hierarchy of labels defined previously. The different bar plot sets correspond to different combinations of the descriptors detailed in sections 4.5.1 and 4.5.2.

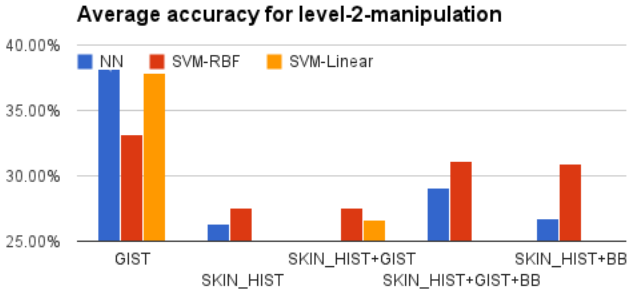
The performance showed that there is a raw accuracy measure, i.e., number of correct tests normalized by the total amount of tests. What gives us an initial idea of how much of a whole sequence we can get to understand, but could be miss-leading because it is not normalized per class, as we saw in the previous example. If we compute the normalized accuracy per class, results in level 1 and level 2-*manip* remain similar, but accuracy normalized per class for level 2-*Non-manip* drops to around 25%. This indi-



(a)



(b)



(c)*

Figure 4.5: Each set of columns represents the average performance (correct classification) for all tests using labels from level 1 (a), level 2 - manipulation (b) and level 2 non-manipulation (c). *NOTE that the percentages are not normalized per class in this plot but per number of tests, therefore results in (c) are miss-leading, since they are actually not better than the other levels, as can be seen in the confusion matrix presented in Table 4.3.

cates that this level classification is not successful with the current image description, what can be analyzed in more detail with the confusion matrices analysis.

Table 4.1: Confusion matrix for labels Manipulation vs Non-manipulation, using *Skin_hist* and *SVM_RBF*.

	Manip	Non-Manip
Manip	**0.84**	0.24
Non-Manip	0.16	**0.76**

Table 4.2: Confusion matrix for fine grained Manipulation labels considered, using *GIST* descriptor and *NN* classifier.

	two_hands	one_hand	pick_up	others
two_hands	**0.24**	0.06	0.20	0.28
one_hand	0.12	**0.44**	0.18	0.11
pick_up	0.54	0.38	**0.45**	0.35
others	0.11	0.13	0.18	**0.26**

See the confusion matrices obtained for these experiments in Tables 4.1, 4.2 and 4.3. We can see that in the last case, the classifier training stage was not successful at all, getting to classify most tests into the same dominant class. As what we observed in the initial (one step) experiment, the low performance is likely to be due to insufficient training: it was properly balanced for certain actions (level 1 and non-manip), but for others with fewer examples it has clearly not enough information to obtain a robust classifier.

Performance using state-of-the-art image representations

In this second set of experiments, we explore more sophisticated, although also more costly, image representation, to evaluate if the issues encountered with more compact

Table 4.3: Confusion matrix for fine grained NON-Manipulation labels considered, using *GIST* descriptor and *SVM-L* classifier. Seat label is not shown because there were no occurrences in this sequence.

	Walk	Stairs	Stand	Screen	Poster	Talk
Walk	0.88	0.99	0.30	0.00	0.61	0.39
Stairs	0.00	0.01	0.00	0.00	0.00	0.00
Stand	0.11	0.01	0.58	1.00	0.28	0.58
Screen	0.00	0.00	0.00	0.00	0.00	0.00
Poster	0.01	0.00	0.12	0.00	0.11	0.03
Talk	0.00	0.00	0.00	0.00	0.00	0.00

Table 4.4: Confusion matrix for labels Manipulation vs Non-manipulation, **using *CNN* descriptor**.

	Manip	Non-Manip
Manip	**0.80**	0.20
Non-Manip	0.12	**0.88**

features were only a matter of feature descriptive power or also the lack of enough training examples.

As a conclusion from this second set of experiments, we compare the best configurations using the proposed compact image description with more sophisticated features in Table 4.7.

We can observe that for the basic classification, the contextual separation between manipulation and non-manipulation, as one could expect, is nicely modeled by our simple description of how the skin (arms-hands) pixels are distributed in the images. How-

Table 4.5: Confusion matrix for labels Manipulation vs Non-manipulation, **using *CNN* descriptor**.

	two_hands	one_hand	pick_up	others
two_hands	**0.61**	0.06	0.22	0.10
one_hands	0.06	**0.63**	0.17	0.15
pick_up	0.14	0.11	**0.42**	0.33
others	0.15	0.20	0.27	**0.38**

Table 4.6: Confusion matrix for fine grained NON-Manipulation labels considered, **using *CNN* descriptor**.

	Walk	Stairs	Stand	Screen	Poster	Talk
Walk	0.99	0.00	0.01	0.00	0.00	0.00
Stairs	0.09	0.91	0.00	0.00	0.00	0.00
Stand	0.87	0.02	0.11	0.00	0.00	0.00
Screen	0.00	0.00	0.00	0.00	0.00	0.00
Poster	0.61	0.38	0.00	0.00	0.02	0.00
Talk	0.92	0.00	0.02	0.00	0.00	0.06

ever, for more complex and fine-grained categorization, the preliminary results we have obtained with the CNN based representation looks like a promising new path for activity recognition

Although these features are more costly to compute, future steps include combining the preliminary results obtained in this work in such a way that the simple per frame classification can be used as a prior or decision step, to select key frames where to evaluate the more detailed representations.

Table 4.7: Accuracy obtained with the best performing options from all the image representations studied. Top rows are global representation. Bottom rows are the results for more sophisticated image representations.

	Level 1	Level 2 Manip	Level 2 Non-Manip*
SKIN-HIST (SVM-RBF)	**0.84**	0.27	0.80
GIST (SVM-L)	0.65	0.35	0.81
BoW	0.81	0.44	0.77
CNN	**0.84**	0.52	**0.83**
CNN-MULTI	0.83	**0.57**	0.77

4.7 Conclusions and Future Work

In this paper, we presented results of our quantitative analysis of different feature extraction methods for the task of activity recognition using an RGB-d wearable vision system. We evaluate a novel and challenging public dataset and propose a hierarchy of labels for the included activities.

Our experiments show that classification in still frames with compact features can give good priors for more sophisticated classifiers/descriptors. Based on our experiments, CNN-based image features provide the best representation for finer grain activity recognition steps, compared to other baselines including the bag of words representation or ad-hoc designed skin histograms, and there is still room for improvement including temporal consistency and increasing the presence of the depth information within the image representation.

Our dataset includes additional wearable cameras that recorded the trajectories simultaneously. Future steps include similar analysis on all cameras to compare their

strengths and weaknesses of each view points for the different tasks.

CHAPTER 5

ANALYZING SEDENTARY BEHAVIOR IN LIFE-LOGGING IMAGES

5.1 Abstract

We describe a study that aims to understand physical activity and sedentary behavior in free-living settings. We employed a wearable camera to record 3 to 5 days of imaging data with 40 participants, resulting in over 360,000 images. These images were then fully annotated by experienced staff with a rigorous coding protocol. We designed a deep learning based classifier in which we adapted a model that was originally trained for ImageNet [18]. We then added a spatio-temporal pyramid to our deep learning based classifier. Our results show our proposed method performs better than the state-of-the-art visual classification methods on our dataset. For most of the labels, our system achieves more than 90% average accuracy across different individuals for frequent labels and more than 80% average accuracy for rare labels.

5.2 Introduction

It is well understood that physical activity (PA) has a significant impact on health. Based on established scientific evidence, the 2008 Physical Activity Guidelines for Americans states that adults should get at least 150 minutes of moderate to vigorous physical activity (MVPA) every week [94]. Recently, new evidence shows that sedentary behavior (SB), or prolonged, unbroken sitting, is a distinct, significant risk factor on health, independent of PA levels [97]. Common examples of SB include television (TV) watching, driving, workplace sitting, and computer use. Insufficient PA and excess of SB are as-

sociated with a considerably increased risk for obesity, type 2 diabetes, cardiovascular disease, depression, and all-cause mortality.

In other words, "too much sitting" is distinct from "too little exercise". Merely meeting public health guidelines on PA is not enough; if one sits for prolonged periods of time, he or she is still exposed to the various metabolic health risks mentioned above.

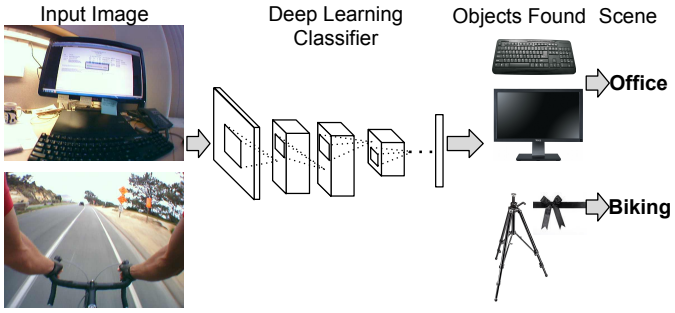


Figure 5.1: Sample result of running deep net object classifier on life-logging images. The objects that are found in top and bottom images are monitor/keyboard and tripod/ribbon. Scene type is predicted based on the object classifications.

Precisely measuring PA/SB is a fundamental, yet non-trivial task. Conventional self-reporting measures are subject to bias and errors. Recent advancements in wearable motion sensors technologies, particularly accelerometer, have enabled objective assessment of PA/SB. Machine-learning techniques have been used to classify everyday activities using annotated motion sensor data (e.g., [3]).

However, it is almost impossible for researchers to identify how exactly PA/SB time is spent with only accelerometer data. A more detailed understanding of the composition of behaviors and their ecological environments will be critical when designing interventions. For example, some SB that periodically occurs at certain periods of time may be less modifiable (e.g., having a family dinner), while some more modifiable (e.g., working at a computer for excessively long, watching TV for 3 hours straight).

In this study, we investigate using a wearable camera (SenseCam [55]) for recording detailed PA/SB in life-logging images. SenseCam is a neck-worn, palm-sized device that automatically takes pictures about every 20-30 seconds. The images objectively capture daily activities in free-living settings and thus become a powerful tool for behavioral scientists to understand specific activity patterns.

After the recent success of deep learning methods in object classification [70], researchers adapted these models for similar object recognition datasets such as Caltech-256 [26] and Caltech-256 [141]. We are going to take this further by applying object classifiers for the purpose of scene classification. We used a model that is pre-trained for object classification and applied it to SenseCam images. In some cases, it finds appropriate objects and sometimes it does not. Fig. 5.1 shows two example queries to the deep learning system. Objects that are detected in the two input images include Keyboard/Monitor and Tripod/Ribbon. Although tripod and ribbon seem to be irrelevant, it is not hard to see the similarities between the scene and these object. We leverage these similarities and make the connection between these predictions and the scenes. For example, prediction of tripod suggests road lines in a biking scene. The idea is somewhat similar to that of ObjectBank [78]. The difference is that the features that we propose to extract are based on a different classifier not specifically trained with related objects and applied at a coarser level.

We collected over 360,000 images from 40 participants over the course of 3-5 days in real-life settings. The manual annotation effort on these images took about 18 man months. To help significantly accelerate this process, we developed a deep learning based method to automatically and efficiently process the images and classifying PA/SB type. Furthermore, we extended the models with temporal and spatial pyramids to improve the performance. Using our novel image representation method, we achieved an

average accuracy of more than 80% for most of the classification labels and more than 95% accuracy for some of the binary classification problems for individual persons.

5.3 Related Work

Image analysis has been researched for decades, but life-logging images pose a number of challenges: (1) the sheer amount of images captured (e.g., thousands per day) demands highly efficient processing algorithms and architectures. (2) the machine learning methods should focus on high-level features for to extract high-level concepts (e.g., using the visual presence of a computer monitor to determine working with a computer).

MyLifeBits [41] was a pioneer project in this research space, but image analysis was not a focus then. More recently, Doherty et al. [22] presented an approach that classified *lifestyle concepts* such as a tree, road, and door in SenseCam images. They used MPEG-7 features: ColorLayout and ScalableColor, the former being a set of DCT coefficients of the YCbCr image and the latter being the color histogram. This low-level feature extraction approach has also been used in their other reported work on life-logging images [5, 11]. We used high-level features instead and as we show in Sec. 5.5, high-level features are more suitable for object recognition (e.g., TV watching, computer use) in understanding PA/SB patterns. Recently, Pirsiavash et al. [104] presented a method for analyzing activities in first-person videos using high-level object-based descriptors.

Our Contributions To the best of our knowledge, we gathered the largest life-logging dataset with reliable annotations. We then designed a deep learning based method to analyze the images. Instead of training a new network from scratch, we used a pre-trained model that was trained for object recognition [26] and showed that with the modifications, it worked better than the variety of the baseline algorithms on the dataset.

We further improved the model by applying spatial and temporal pyramids.

5.4 Classification Methods

In this section, we describe the image representation methods we propose to build our classification framework. Our goal is to have a diverse and powerful set of baselines to compare with our proposed deep learning based method.

Bag of Visual Words (BOW) Bag of visual words is a popular classic visual classification method that we used as a baseline in our experiments. We used dense SIFT [80] to extract interest points and used k-means clustering to create a codebook of visual words. Then each image was represented by a histogram over these visual words which was implemented with a fast kd-tree search data structure.

Fisher Encoding Fisher encoding has shown its promise on some popular object recognition and detection datasets such as PASCAL VOC, Caltech 256 [100] and ImageNet Large Scale Visual Recognition Challenge [115]. We encoded the same features that we extracted for the BOW method with Fisher Encoding.

Color For color representation, we used the same framework as our BOW method with the difference that we used RGB values for each point instead of SIFT features. For this, we created a bag of RGB points and clustered the points into 1000 clusters which were then used to create histogram representations.

GIST GIST is a holistic image feature proposed by Oliva et al. [96] based on a set of Gabor filters. We used 24 kernels at 3 scales separately applied to the RGB channels.

Deep Learning In 2012, Krizhevsky et al. presented a deep learning architecture that achieved the best results on the largest annotated object recognition dataset (ImageNet) [70]. Their model consists of 5 convolutional and 3 fully connected layers with an additional set of pooling and normalization layers. Recently, Girshick et al. achieved state-of-the-art results on PASCAL VOC detection dataset by using a network that is trained to classify ImageNet categories [43]. We claim that the ImageNet deep network is not only good for object recognition purposes but also for obtaining state-of-the-art results on non-object centric data such as our life-logging dataset.

We use a model that is trained to classify ImageNet 2012 categories [26] and run feed forward passes on our SenseCam images and use the outputs of the last layer to represent the input image. The last layer shows the likelihood of assigning the input to the 1000 object classes used to train the network. The representations are used to train a linear SVM model. To represent a smaller object, we ran the feed forward path on multiple windows i.e. the whole image, four corners and center windows. We concatenated the outputs of the deep net to represent the image.

Classification We train linear support vector machines (SVM) [29] to classify different types of features that we extract from the images. Sec. 5.5.2 discusses the implementation details to address the time efficiency issues and explain why we use linear SVMs. In the test phase, given a linear SVM model, one only needs to compute a dot product between the SVM weight vector and the feature vector and use a threshold on the dot product value to determine a specific label.

Multi-frame encoding The pipeline that we have described so far is based on the single frame representation. To capture the dynamic changes in the image sequence, we propose to use multiple frames as input to the classification. To encode the camera at a certain time, in addition to the closest frame to that time, we encode k frames before

and after and concatenate the encoded feature vectors. This simple yet effective method is capable of detecting the amount of change in the image sequence.

The basic idea behind feature concatenation is that instead of detecting the change using low-level features, it is performed at the highest level with the encoded features. When there is no change between a frame and the frames before and after it (i.e., scene is stationary), the encoded features for those frames become similar. By examining the differences between elements of the final feature vector, a linear classifier can detect when a change occurs. However, the change does not have to be detected with a separate classifier as change detection can be merged with the classification of different labels. Thus, apart from focusing on the visual appearance of the scene, meanwhile, the classifier can detect change or motion which is essential for detecting some of the labels.

Parallelization Techniques The main design goal for our system is to be able to process large amounts of data. For example, dense feature extraction on all of the images in the dataset produces more than 600 GBs. This size makes it hard to process on a single machine. As mentioned above, the learning phase has 4 steps: feature extraction, codebook learning, encoding, and training classifier where the testing phase has only 3 steps which exclude the codebook learning.

5.5 Experiments

5.5.1 Data Acquisition and Annotation

We chose to use SenseCam, a wearable device built by Microsoft Research. It is a mature technology and has been used in many other research studies. The participants

were adult cyclists recruited through a university-based cycle-to-work network. Eligible participants were aged 18–70 years, were university employees, routinely bicycled for transportation, and were able to complete surveys in English. Participants provided informed consent and agreed to wear the SenseCam devices during waking hours for 3–5 days. Approximately 50% of them wore the device during the weekend, and on at least one work day; the remainder wore the unit on weekdays only. They were instructed to perform their normal daily living activities, and turn off the SenseCam in private time (e.g., in the bathroom). Excluding night time or private pictures that were removed before our experiments, our dataset contains over 360,000 images.

We recruited staff to manually code these images using a rigorous protocol to ensure annotation reliability and accuracy. The annotation was performed using the SenseCam CLARITY browser. The images were annotated by position labels (i.e., Sitting, Standing Still, Walking/Running, Biking), and activity labels (e.g., Household Activity, Administrative Activity, Television, Other Screen Use and Eating). To the best of our knowledge, this becomes the largest SenseCam dataset with reliable labels.

Dataset Sharing Because of the sensitivity of the images, we only share extracted features to allow reproduction of the experiment results. Instructions on how to access the dataset is available on our lab's website[1]. Also, we provide the service to run new feature extractors upon request.

5.5.2 Results

Timing Feature extraction, encoding and classification are designed to be distributed processes. For feature extraction and encoding, we divided the task into 1000 distributed

[1] http://vision.ucsd.edu/%7emohammad/sensecam_dataset

processes. Each process took about 5 and 3 minutes respectively on our computing cluster. That is equivalent to a total of 3.5 and 2 days of computing. The jobs finished in less an hour because the jobs were running in parallel on multiple machines. The codebook learning step took about 15 hours on one machine using randomly sampled feature points and SVM training took less than 2 minutes for each label and person. The SVM learning took about 1 day of combined time.

As stated earlier, we defined binary classification problems where the labels for training images were known and the classifier was expected to predict the labels for the new unseen testing images. We split the training and testing images with a uniform random distribution with a fixed proportion of training examples.

Our pilot experiments showed that SVMs performs better than other classifiers such as nearest neighbor classifier. We used liblinear [29] which can generate scores along with the binary predictions. The generated scores are in fact the inner product of the input feature vector with the learned weight vector. The pair of weight vector and offset (threshold) define the linear classifier. Having a fixed learned weight vector, we generated the ROC curve for the classifier by sweeping the threshold.

In all of our experiments, we trained and tested binary classifiers on individual persons' data and generated an average ROC curve along with each label. To do this, 40 different classifiers were trained and tested on the two different subsets of each person's data. The measure we use to quantify the ROC curves is normalized accuracy. We normalized the accuracies based on the distribution of the test labels to be able to compare the performance of our classification method with respect to different labels. Note that a random classifier performs with 50% normalized accuracy even when there is a large unbalance between the two classes.

Table 5.1: This table compares the accuracy of different methods on our dataset. *stand st* and *stand mv* represent standing still and standing/moving.

	sitting	stand st	stand mv	walk/run	biking
bag of words	79.9	69.7	62.0	73.3	92.8
fisher	86.8	76.2	76.3	83.2	94.4
decaf	**93.4**	76.8	83.3	**89.9**	97.1
pyramid decaf	92.8	**80.3**	**85.3**	89.7	**98.5**
bag of color	82.6	62.0	64.0	83.6	95.3
gist	75.2	62.9	62.7	74.1	90.1

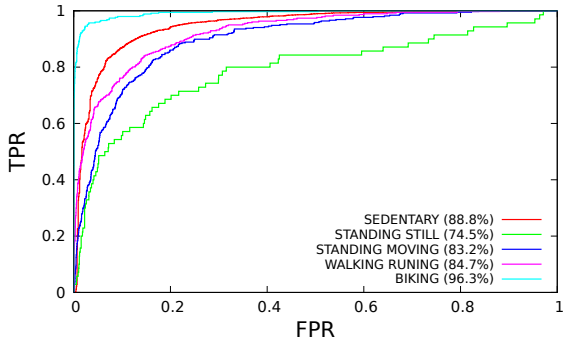


Figure 5.2: ROC curves for classification (decaf) of different position labels along with their average normalized accuracies.

Fig. 5.2 shows the ROC curve of our method (deep learning, spatial and temporal pyramids) for different position labels. It shows that sitting and biking achieved the best results with average accuracy of 88.8% and 96.3% respectively. Standing Still showed the worst performance with the average accuracy of 74.5%. The main reason is that there were not enough samples for the Standing Still class, however, the raw accuracy was more than 99%. Table 5.1 compares our methods (*decaf* and *decaf_pyramid*) with other baselines. As the table shows, *decaf* worked the best for 2 labels and *decaf_pyramid* for the other 3.

Fig. 5.3 compares our multi-frame coding approach with single frame coding. In this case, the average accuracy improved from 89.4% to 93.4%. In these experiments, sedentary label was chosen for illustrative purposes. For this label, using 7 frames which is equivalent to 2–3 minutes results in the best classification accuracy. Other labels have different optimal points, but they are all about 7 frames. Using multi-frame coding improves accuracy by about 3–4% in average.

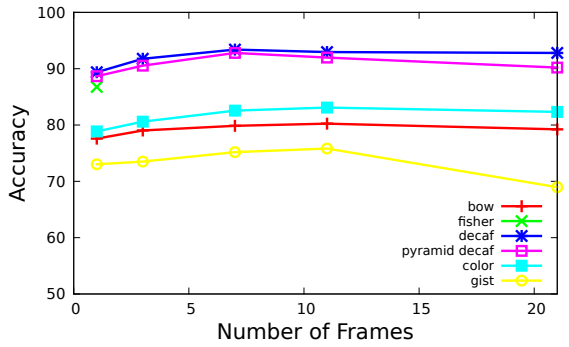


Figure 5.3: The effect of changing the number of frames in our multi-frame coding method. The sitting label is used for this experiment.

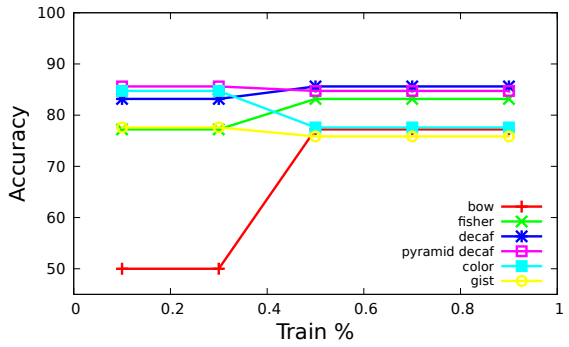


Figure 5.4: The effect of changing the number of images used for training.

Fig. 5.4 shows the effect of changing the amount of training data. Most of the curves are almost flat which shows that the same accuracy can be achieved using as little as 10% of the training the classifier. This shows that the classifier can generalize well even with small amount of training data. In this experiment, the test set shrinks as the training set grows.

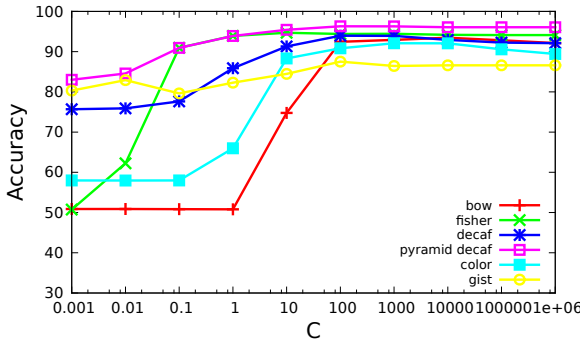


Figure 5.5: The effect of changing the SVM regularization term.

Fig. 5.5 shows the effect of the regularization term $\frac{1}{C}$ in the SVM classification. Our experiments showed that choosing low regularization term (i.e. high C) improves the overall accuracy. The reason is the imbalance between the two classes. With high regularization parameter, the model tends to overfit to the dominating class and therefore it reduces the accuracy. Based on these experiments, we use $C = 100$ in all of our experiments.

5.6 Conclusions

In recent years, there has been a surge of interest in wearable cameras. This presents a need for efficient and accurate image analysis techniques for processing life-logging images. We presented a deep learning based classification method built on top of an ImageNet classifier and showed that it performs better than all baselines. Furthermore, we designed a spatial pyramid technique in time and space to further improve the classification results. We achieved more than 80% average accuracy for most of the labels and more than 90% accuracy for classification of some of the individuals' data.

CHAPTER 6

BOOSTED CONVOLUTIONAL NEURAL NETWORKS

6.1 Abstract

Deep convolutional neural networks (CNNs) have achieved promising performance on a diverse range of visual recognition tasks by automatically learning image representations. One of the most challenging problems in developing large, deep CNNs is the training procedure. In this paper, we propose a novel model called *Boosted Deep Convolutional Neural Networks* to train huge networks efficiently by combining the merits of boosting and deep CNNs. Our proposed approach achieves state-of-the-art results on standard fine-grained image classification datasets.

6.2 Introduction

Deep convolutional neural networks (CNNs) have recently produced outstanding results in learning image representations for several vision tasks including image classification [71, 131, 53] and object detection [44, 56, 108]. These neural networks usually consist of several components including convolutional, fully connected and pooling layers. By stacking several of these layers, deep CNNs are capable of learning complex features that are highly invariant and discriminant [142, 71, 121, 128, 122]. Krizhevsky et al. [71] proposed an eight layer deep network architecture that produced state-of-the-art performance on the ImageNet Challenge [111]. Given the success of this network, it has been applied widely to many other problems such as video classification [64], face recognition [129] and action recognition [91]. However, the optimal image representation for

each computer vision task is unique and finding the optimal deep CNN structure for extracting that representation is a challenging problem. There are some general guidelines, inspired by biological vision systems, for designing these deep networks such as putting convolutional layers early in the network. These guidelines, however, do not address how to set structural parameters of the networks such as the number of layers, number of units in each layer or the size of the receptive fields in the pooling layers. As a result designing a new network can take weeks of trial-and-error.

To address this design challenge, we propose to combine boosting and deep CNNs. The idea is to leverage the ability of boosting to combine the strength of multiple weaker learners to simplify the complicated design process of deep CNNs. This characteristic of boosting has been shown to be very significant for several Computer Vision tasks. For example in the Viola and Jones face detector [135] boosting searches a very large pool of simple classifiers (thresholds on Haar wavelet responses) and trains a classifier that is able to detect a complex object such as a human face. Our approach is also motivated by the successful combination of decision trees [105] and boosting. As in the case of CNNs, the design of decision trees is not straightforward. What is the optimal depth of a tree? How many branches should there be per node? What are the optimal criteria for splitting tree nodes? These are all important aspects of learning a decision tree and much research has been devoted to these questions in the 1990s. For example, there are several proposals for node splitting criteria such as Gini impurity, information gain [88] and there are several tree induction algorithms such as CART [7] or C4.5 [106]. Today, boosted decision trees [36, 37, 38] have eliminated most of these problems via an optimally weighted vote over decision trees that are individually sub-optimal. Similar to the success story of boosted decision trees, we believe that combination of boosting and deep learning can significantly reduce the challenges in designing deep networks.

Another motivation for this proposal is that in most cases, e.g., [71, 121], researchers have to train multiple CNNs and average their results to obtain the best performance. This technique is a special case of bagging [8] which has been shown to have inferior performance when compared to boosting [4]. The advantage of boosting over bagging is intuitive as instead of training multiple independent networks, it trains each new network by focusing on the mistakes of the current committee of networks.

The idea of boosting neural networks or, more generally, working with ensembles of neural networks has been around for many years; see for example [28, 27, 120, 118, 119, 118, 144, 46, 15, 14, 2, 63]. All these works demonstrated advantages of using an ensemble of networks over using a single large network. These works, however, either rely on simple averaging of several networks or rely on some heuristic weighting mechanism to impose boosting weights in the training process. In addition, some of these methods do not scale to the object recognition tasks that pervade the modern computer vision literature.

In this work, we propose a new algorithm for boosting deep networks (BoostCNN) to combine the merits of boosting and deep CNNs. To learn this new model, we propose a novel algorithm to incorporate boosting weights into the deep learning architecture. The proposed BoostCNN method can outperform standard techniques for learning deep CNNs on a variety of standard fine-grained classification datasets. Also, BoostCNN gives us the ability to incorporate different CNNs architecture within a unified framework.

6.3 Multiclass boosting

We start with a brief overview of multiclass boosting. A multiclass classifier is a mapping $F : \mathcal{X} \to \{1 \dots M\}$ that maps an example x_i to its class label $z_i \in 1 \dots M$. Since this is not a continuous mapping, a classifier $F(x)$ is commonly trained through learning a *predictor* $f : \mathcal{X} \to \mathbb{R}^d$ for some d. The classifier $F(x)$ is then implemented by

$$F(x) = \arg \max_{k=1\dots M} \langle y_k, f(x) \rangle, \tag{6.1}$$

where y_k is a unit vector that represents label of the k^{th} class and $\langle \cdot, \cdot \rangle$ is the dot product.

For example in binary classification, labels are $y_1 = +1$ and $y_2 = -1$ and (6.1) is equivalent to the popular $F(x) = sign[f(x)]$ decision rule. Another example is one-vs-all multiclass classifiers. In this method, for each class k, a predictor $f_k(x) : \mathcal{X} \to \mathbb{R}$ is trained to discriminate between examples of that class versus others. In order to classify a new example $\hat{x}$, $f_k(\hat{x})$ is computed for all $k = 1 \dots M$ and the class of largest predictor is declared as the label. This procedure is equivalent to (6.1) by defining $f(x) = [f_1(x) \dots f_M(x)] \in \mathbb{R}^M$ and $y_k = \mathbf{1}_k \in \mathbb{R}^M$, i.e. k^{th} element is one and the rest are zero. In general, the choice of labels are not restricted to the canonical basis in $\mathbb{R}^M$ and it is possible to use any set of M distinct vectors $y_1 \dots y_M \in \mathbb{R}^d$ where $d > 2$ [114]. For simplicity, however, in the rest of this paper, we assume that $d = M$ and $y_k = \mathbf{1}_k$.

Multiclass boosting is a method that combines several "multiclass predictors" $g_i : \mathcal{X} \to \mathbb{R}^d$ to form a strong committee $f(x)$ of classifiers, i.e., $f(x) = \sum_{t=1}^{N} \alpha_t g_t(x)$ where g_t and α_t are the weak learner and coefficient selected at t^{th} boosting iteration. There are several approaches for multiclass boosting such as [89, 114, 58]. We use the GD-MCBoost[114].

GD-MCBoost trains a boosted predictor $f(x)$ by minimizing risk of classification

$$\mathcal{R}[f] = E_{X,Z}\{L(z, f(x))\} \approx \sum_i L(z_i, f(x_i)), \tag{6.2}$$

$$\text{where} \quad L(z_i, f(x_i)) = \sum_{j=1, j\neq z_i}^{M} e^{-\frac{1}{2}[\langle y_{z_i}, f(x_i)\rangle - \langle y_j, f(x_i)\rangle]}, \tag{6.3}$$

via gradient descent in function space. GD-MCBoost starts with $f(x) = \mathbf{0} \in \mathbb{R}^d \ \forall x$ and iteratively computes the directional derivative of the risk to update $f(x)$ along the direction of $g(x)$

$$\delta\mathcal{R}[f; g] = \frac{\partial\mathcal{R}[f + \epsilon g]}{\partial\epsilon}|_{\epsilon=0} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{M} g_j(x_i)w_j(x_i), \tag{6.4}$$

where we assumed $y_j = \mathbf{1}_j$ and

$$w_k(x_i) = \begin{cases} -e^{-\frac{1}{2}[f_{z_i}(x_i) - f_{y_k}(x_i)]} & k \neq z_i \\ \sum_{j=1|j\neq k}^{M} e^{-\frac{1}{2}[f_{z_i}(x_i) - f_{y_k}(x_i)]} & k = z_i \end{cases} \tag{6.5}$$

GD-MCBoost then selects/trains a weak learner g^* that minimizes (6.4), e.g.

$$g^* \quad = \quad \arg\min_{g\in\mathcal{G}} \delta\mathcal{R}[f; g], \tag{6.6}$$

and compute the optimal step size along g^*,

$$\alpha^* = \arg\min_{\alpha\in\mathbb{R}} \mathcal{R}[f + \alpha g^*], \tag{6.7}$$

using a line search. The boosted predictor $f(x)$ is finally updated as

$$f = f + \alpha^* g^*. \tag{6.8}$$

6.4 Boosting convolutional neural networks

In this work, we propose to combine the merits of boosting and deep CNNs. This proposal is motivated by the successful combination of boosting and decision trees [37, 38] which has significantly simplified design of decision tree classifiers.

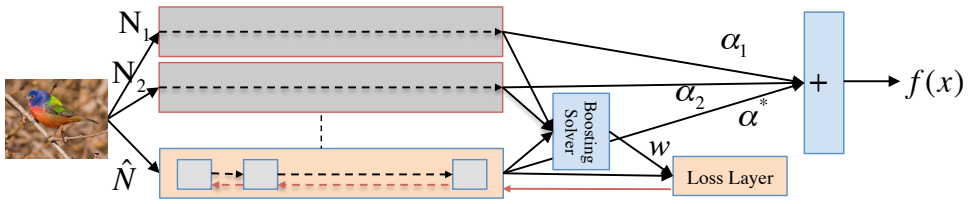


Figure 6.1: The proposed architecture for boosting CNNs.The yellow network at the bottom is the network that being learned and light gray networks show the previously trained CNNs.

Training a CNN to minimize the boosting loss. In this subsection we describe our proposed method to train a network that minimizes (6.4). Assume that after t iterations of boosting an ensemble of t networks $\mathcal{N}_1, \dots \mathcal{N}_t$ are trained and the current boosted predictor is

$$f(x) = \sum_{\mathcal{N}_t \in \mathcal{A}} \alpha_t \mathcal{N}_t(x). \tag{6.9}$$

According to Algorithm 1, the optimal network to add the ensemble in iteration $t+1$ is the network that minimizes (6.4). However, the learning algorithms for convolutional networks is based on minimizing the error rate, e.g. log-likelihood which is independent of the boosting weights and can be very different from (6.6). To train such a network, one possibility is to replace the log-likelihood loss function with the (6.4) in the back-propagation algorithm. However (6.4) is unbounded as scaling $\mathcal{N}(x)$ can make it infinite. In practice, direct optimization of (6.4) makes the learning process unstable as the loss diverges quickly.

In order to address this issue, we first note that (6.4) is a summation of dot products between the network output $\mathcal{N}(x_i)$ and boosting weights $w(x_i)$. Therefore (6.4) measures the similarity between those vectors and thus the optimal network output, $\mathcal{N}(x_i)$, has to be aligned with the boosting weights, i.e.,

$$\mathcal{N}(x_i) = \beta w(x_i), \tag{6.10}$$

where $\beta > 0$. Note that the exact value of β is not crucial during the weak learner training process because $\mathcal{N}(x)$ will be scaled appropriately by the optimal α (6.7) after the training. Therefore without loss of generality we can assume that $\beta = 1$ and the optimal network output has to replicate boosting weights. This is equivalent to train a network $\mathcal{N}(x) = [n_1(x) \dots n_M(x)] \in \mathbb{R}^M$ to minimize the square error loss

$$\mathcal{L}_{se}(w, g) = \sum_{i=1}^{n} \sum_{j=1}^{M} (n_j(x_i) - w_j(x_i))^2. \tag{6.11}$$

This criterion for learning weak learner network is one of the main differences between this work and previous works such as [120], where they, instead, minimized weighted error rate.

Using $\mathcal{L}_{se}$ loss function for learning a CNN, the back-propagated derivatives are

$$-\frac{\partial \mathcal{L}_{se}}{\partial n_k(x_i)} = 2(w_j(x_i) - n_k(x_i)). \tag{6.12}$$

Computing the boosting weights. According to Algorithm 1, in each iteration of boosting, weight (6.5) has to be computed for each of the training examples. According to (6.5) the weight of each example is proportional to how well it is classified by the current committee of weak learners. If an example is correctly classified, i.e. $g_{z_i}(x_i) > g_k(x_i)\ \forall k \neq z_i$, then all exponential terms in (6.5) will be small. This reweighting procedure is an essential part of boosting algorithm as it enables boosting to train new weak learners on the weaknesses of the current boosted classifier.

After each iteration of boosting, we calculate the boosting weights for all of the training examples. For this purpose, we need to compute f_{y_j} which is a very costly process. Instead we can rewrite (6.5) to reuse previously calculated weights ($w'_k(x_i)$). Based on (6.13), we only need the latest boosting weights and the output of the last network to calculate boosting weights for the next round.

$$w_k(x_i) = \begin{cases} -e^{-\frac{1}{2}[g_{z_i}(x_i) - g_{y_k}(x_i)]} w'_k(x_i) & k \neq z_i \\ -\sum_{j=1|j\neq k}^{M} w_k(x_i) & k = z_i \end{cases} \tag{6.13}$$

Boosted classifier. Next, the $\mathcal{L}_{se}$ loss function layer uses boosting weights and output of last network, $\widehat{\mathcal{N}}$, to compute the derivatives of (6.12) and back-propagates it to the internal layers of $\widehat{\mathcal{N}}$. This process iterates until $\mathcal{L}_{se}$ converges and $\widehat{\mathcal{N}}(x)$ is trained. Then, we need to estimate the optimal boosting coefficient $\hat{\alpha}^*$ by (6.7). We employ a binary search technique to solve $\frac{\partial \mathcal{R}[f+\alpha g^*]}{\partial \alpha} = 0$. Finally the boosting coefficient α^t is updated as $\alpha^t = \nu \hat{\alpha}^*$ where $\nu \in (0, 0.1]$ is the shrinkage parameter which has been shown to be effective for stochastic boosting algorithms [38].

6.5 Analysis

The proposed BoostCNN algorithm has a couple of interesting properties. We start by providing some intuition about the boosting weights and their effects on training CNN learners, e.g., $\widehat{\mathcal{N}}$ in Figure 6.1.

According to (6.5), the boosting weights in BoostCNN algorithm are M-dimensional. These weights encode two type of information. First, the norm of vector $w(x) \in \mathbb{R}^M$ is proportional to how well example x is classified by the current ensemble of weak learners. If x is correctly classified then $f_{z_i}(x_i)$ will be larger than $f_k(x_i)$ $\forall k \neq z_i$, the terms in the exponents of (6.5) will be small and thus $w(x)$ will have a small norm. On the other hand, if x is mis-classified, some of the exponent terms in (6.5) will positive and $w(x)$ will have larger norm. Second, the k^{th} components of vector $w(x_i) \in \mathbb{R}^M$, encodes the importance of k^{th} class in classification of example x_i. For an incorrect class label $k \neq z_i$, if $f_k(x_i) > f_z(x_i)$ then $w_k(x)$ will be large. In addition $w_k(x)$ will increase

Algorithm 1: **BoostCNN**

Input: number of classes M, number of boosting, iterations N_b, number of iterations for learning a CNN N_c, shrinkage parameter ν, and dataset $\mathcal{D} = \{(x_1, z_1), ..., (x_n, z_n)\}$ where $z_i \in \{1 \dots M\}$ is label of example x_i.

for $t = 1$ to N_b **do**

  for $l = 1$ to N_c **do**

    sample a batch of training examples.

    run a forward pass for network $\widehat{\mathcal{N}}$.

    back-propagate $2(w(x) - \widehat{\mathcal{N}}(x))$ to the layers of $\widehat{\mathcal{N}}$.

  end for

  compute $w(x_i)$ for all x_i, using (6.5)

  find $g^*(x)$, α^* using (6.6) and (6.7)

  update $f(x) = f(x) + \alpha^* g^*(x)$

  set $\mathcal{N}_t := \widehat{\mathcal{N}}$.

  set $\alpha_t := \nu\alpha^*$

end for

Output: predictor $f(x) = \sum_{t=1}^{N_b} \alpha_t \mathcal{N}_t(x)$

exponentially by increasing $f_k(x_i) > f_z(x_i)$. Therefore weights of incorrect classes will get magnified exponentially. Similarly, if $f_k(x_i) < f_z(x_i)$, $w_k(x_i)$ will be a small value. These two weighting mechanisms will modulate the network output $\widehat{\mathcal{N}}(x)$ in (6.4) and help CNN learning procedure to focus on more difficult examples and more confusing classes.

Another interesting property of CNN-Boost method is that, for any network $\bar{\mathcal{N}}$ for which (6.4), i.e. $\sum_{i=1}^{n}\sum_{j=1}^{M} \bar{\mathcal{N}}(x_i)w_j(x_i)$, is non-zeros, adding $\bar{\mathcal{N}}$ (or $-\bar{\mathcal{N}}$) can help the

gradient descent procedure to further minimize the classification risk of the boosted classifier, (6.2) and improve the performance. In fact if $\mathcal{N}'(x)$ is network with uniform random output, i.e.

$$Prob(k = \arg\max_{k=1...M} \mathcal{N}'(x)) = \frac{1}{M}, \tag{6.14}$$

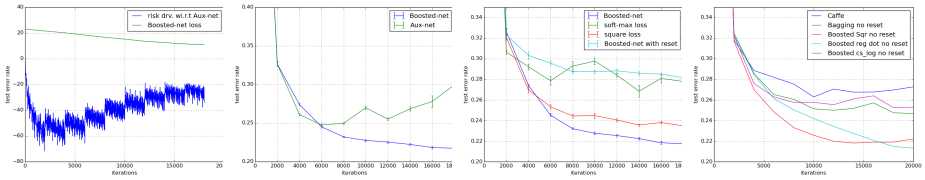
then according to (6.4) and (6.5)

$$\begin{aligned} E\{\delta\mathcal{R}[f;\mathcal{N}']\} &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{M} E\{g_j(x_i)\}w_j(x_i) \\ &= -\frac{1}{2M}\sum_{i=1}^{n}\sum_{j=1}^{M} w_j(x_i) = 0. \end{aligned} \tag{6.15}$$

Therefore, BoostCNN can utilize any network whose output is slightly better than random for boosting. This is very significant as it can reduce the trial-and-error that is required for designing stand-alone CNNs.

The proposed network architecture in figure 6.1 is similar to multi-column deep network [2, 14] and averaging several learning networks [71, 121]. Multi-column CNN trains a group of CNN's simultaneously to learn a linear combination of these network as the final predictor. This, however, will increase the complexity of the learning process as it will exponentially increase the number of local minima in the network optimization problem, e.g. any permutation of columns of a local optimum is also a local optimum. Comparing this method with the proposed method, in BoostCNN 1) networks are trained sequentially and 2) each network is trained on the mistakes of the previous networks. This sequential network learning simplifies the optimization problem and avoids the local minima problem of multi-columns networks. Similarly, BoostCNN is better than averaging several independently trained networks because BoostCNN optimizes coefficients of the linear combination and trains new networks on more difficult examples and classes.

Finally, note that for training $\widehat{\mathcal{N}}$ in each boosting iteration it is possible to initialize it



(a) (b) (c) (d)

Figure 6.2: (a) loss of boosted network (6.3) and risk derivatives of the last network, (6.4), on the training set, (b) accuracy of boosted network and last network $\widehat{\mathcal{N}}$ during the training, (c) accuracy of the classifiers trained with BoostCNN and other methods on the test set., (d) comparison between bagging and boosting.

by random parameters or by parameters of $\widehat{\mathcal{N}}$ of the previous network. According to our experiment, the later is more effective and we will further discuss this issue in section 6.6.

6.6 Experiments

In this section, we illustrate properties of the proposed BoostCNN algorithm and compare its performance with the standard deep learning method on several image classification tasks. For implementation, we used the open source Caffe framework [59] deep learning library.[1]

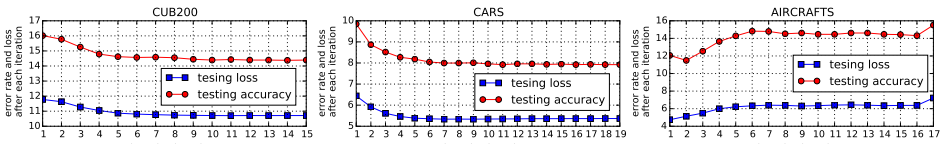
We start by illustrating properties of BoostCNN using CIFAR-10 dataset [69]. For training, we used CIFAR10-quick network model provided by the library which consists of three convolutional layers, each followed by a pooling layer and a rectified linear unit (RELU). These layers are then augmented by two fully connected layers.

Using BoostCNN, we trained an ensemble of 9 networks each with $2,000$ back-propagation iterations. Figure 6.2-(a) shows evolution of 1) functional risk derivative of

[1] Our open source code will be available on our project website.

(6.4) with respect to the output of the current network $\widehat{\mathcal{N}}$ and 2) risk of classification, (6.3), for the boosted CNN classifier. First, note that the sharp jumps in the figure are corresponding to start of a new boosting iteration. Second, $\widehat{\mathcal{N}}$ starts with random parameters and its corresponding risk derivatives, (6.4), is close to zero. By back-propagating derivatives of the square error loss function (6.11), BoostCNN then improves this network to minimize (6.6) and trains a classifier. Third, at the end of the first boosting iteration, the trained network, $\widehat{\mathcal{N}}$, will be stored as $\mathcal{N}_1$. This will change the boosted classifier and the boosting weights resulting in a jump in (6.4). This is not surprising as the current $\widehat{\mathcal{N}}$ is optimized for the old boosting weights and it is sub-optimal with respect to the new weights. In the second boosting iteration, BoostCNN then starts fine-tuning $\widehat{\mathcal{N}}$ to optimize it for the new weights and minimize (6.4). Fourth, the risk derivatives, (6.4), at the start of each boosting iteration increase as boosting progress. This is because as the boosting algorithm progresses the boosted classifier becomes better and it becomes harder to train a CNN weak learner to address its mistakes. Finally, Note that in the proposed method we initialized $\widehat{\mathcal{N}}$ with the parameters of the last trained $\widehat{\mathcal{N}}$. If instead, we choose to randomly initialize these networks the jumps will be higher because according to (6.15) for a random network (6.4) is close to zero.

Figure 6.2-(b) shows the error rate of the boosted CNN and the last network. As expected the error of the Boosted network decreases by adding more CNNs. On the other hand, the error rate of last network starts increasing after $4-5$ boosting iterations. The reason is that the last network is trained to optimize (6.4) which can be very different from accuracy of the network. For example in the first boosting iteration of BoostCNN $f(x) = 0 \in \mathbb{R}^{10}$ and $w_j(x_i) = -1$ for $j \neq z_i$ and $w_{z_i}(x_i) = 9$. Using these weights, optimizing (6.4) is equivalent to minimizing error rate for *all* training example. However, in the last boosting iterations many examples are correctly classified and receive lower weights which let the network to focus on the harder examples.



(a) (b) (c)

Figure 6.3: This figure shows how accuracy and loss change after each boosting iteration.

Regarding accuracy, figure 6.2-(b) compares the error rate of boosted CNN predictor with the error rate of a CNN trained with soft-max loss function, and a CNN trained with square error loss function, (6.11). As shown in this figure, the predictor of boosted-CNN achieved lower error. We also continued the training of non-boosted networks for $40K$ iterations, but error rate did not improve significantly. Manually changing the learning rate of the soft-max network will decrease the error to around 25% which is still 3% higher than the error rate of boosted CNN. Finally, this figure also illustrates the performance of a boosted CNN predictor when parameters of the last network are reset after each boosting iteration. As shown in this figure, the initialization technique for the last network significantly improves the performance of the boosted CNN predictor.

Bird Species Classification. CUB-200-2011 [136] consists of $11,788$ images of 200 bird species. This dataset comes with a pre-selected training and testings splits. We cropped the largest square window from the center of all images and resized them to 448×448 in this experiment. We don't consider random data alterations in the classifier. However, we mirrored the training images and doubled the training set. We use B-Net in this experiment and initialize the last layer using externally trained Linear SVM to reduce the total training time. The Linear SVM is equivalent to the softmax loss used in the network. However, we found out that learning the last layer outside of the network to initialize training leads to more stable classifiers and better results. Note that this is only possible when dealing with relatively small datasets, otherwise computing all

of the features requires a significant amount of memory. In this experiment we used B-CNN [D, D] (B-Net) [131] as the base classifier and we achieved 14.4% error rate vs. the state-of-the-art 15.9% [131]; see Fig. 6.3(a).

Car make and model classification The car dataset [68] consists of 16, 185 images of 196 different car make and models from Acura RL to Volvo XC90. This dataset comes with a predefined training and testing splits. The images include a variety of sizes and aspect ratios and we cropped the largest square window and resized it to 448×448 for pre-processing. BoostCNN achieves the error rate of 7.9% which is only 0.5% higher than the state of the art [68], however the employed 3D for this task which is beyond our current model.

Aircraft Classification. FGVC-aircraft [84] consists of 10k images of 100 different aircraft models. There are different sub-models of the same aircraft design included such as different Boeing 737s. We resized the images to 448×448 while ignoring the original aspect ratio for this experiment and similar to other datasets we double the training set by mirroring. Then we used B-Net as our base classifier from [131] and ran 20 iterations of boosting. Figure 6.3 shows that the testing error rate decreased from 12.1% to 11.5% after which it increases. The reason is that we used a fixed 0.1 shrinkage parameter for all of our experiments. However, this particular classifier needs a lower shrinkage parameter. This is clearer when analyzing the training loss.

6.6.1 Boosting heterogeneous classifiers

So far in this paper, we used homogeneous classifiers in training the boosted classifier. However, BoostCNN is not limited to homogeneous classifiers. We performed an experiment wherein each boosting iteration instead of optimizing one network, we

Method	Accuracy
BoostCNN	85.6%
BoostCNN(heterogeneous)	**86.2**%
Bilinear CNN [131]	84.1%
Krause et. al [67]	82.0%
Pose Normalized CNN [6]	75.7%
Part-based RCNN[143]	73.9%

Method	Accuracy
BoostCNN	92.1%
Bilinear CNN [131]	91.3%
Krause *et al.* [67]	**92.6**%
Chai *et al.* [12]	78.0%
Fisher Vector [45]	82.7%

Method	Accuracy
BoostCNN	**88.5**%
Bilinear CNN [131]	84.1%
Fisher Vector [45]	80.7%
Chai *et al.* [12]	72.5%

Table 6.1: CUB, CARS and AIRCRAFTS results

trained multiple networks and let them compete to achieve the best performance. At each boosting iteration, we train these CNNs independently to minimize (6.4). Then the best classifier is selected based on the overall training loss and is added to the boosting set.

The classifiers we designed for this experiment were B-Nets with 8 different input image variations. We experimented with two different input sizes 448 and 224, two aspect ratios 1 and 1.25 and the choice of cropping the center patch and resizing the image to the desired size. Note that B-Nets work with any input size since the output of the bilinear layer only depends on the number of channels.

Figure. 6.4 shows the result of this experiment. Each column shows the testing accuracy after each boosting iteration and each row corresponds with a different classifier. The recipe that proposed by Lin et. al [131] for CUB is to crop the center patch and resize it to 448×448 which is on par with the 84.1% at first column in the second row. However, cropping the center rectangle of size 560×448 seems to help in the later stages of boosting. Figure. 6.4 shows that combination of different base classifiers outperform boosting only one type of classifier (Figure. 6.3-(a)) as those other classifiers see slightly different perspectives of the input. This boosted classifier converges to 86.2% testing error after 25 iterations which boosting only B-Nets on center 448 square patches converges to 85.6% according to Figure 6.3.

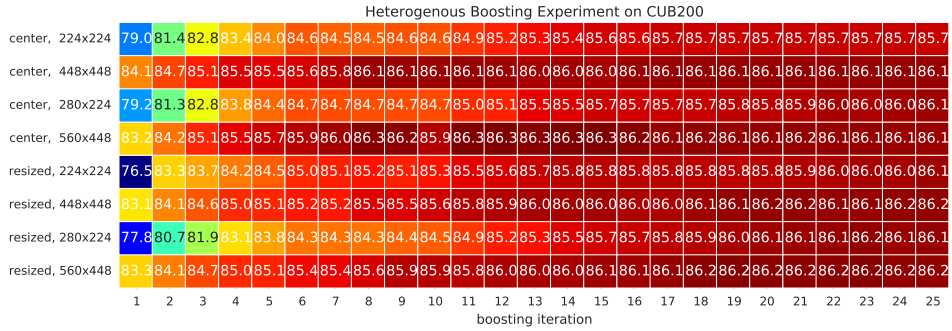


Figure 6.4: Testing accuracy on CUB for 25 boosting iterations using 8 different boosted classifiers.

6.7 Conclusion and future work

In this paper, we proposed a novel model by combining the merits of boosting and deep CNNs. We are inspired by the powerful image representation learned by deep CNNs and the ability of the boosting to combine the strengths of multiple learners to improve the classification. To learn this new model, we developed an algorithm to incorporate boosting weights into the deep learning architecture. We illustrated the properties of boosted convolutional networks and demonstrated the advantages of our model via state-of-the-art results on several fine-grained classification datasets: CUB [136], Cars [68] and Aircrafts [84]. In future work, we will apply BoostCNN to general large-scale classification datasets, e.g., ImageNet. We will extend our heterogeneous boosting experiments to include more diverse networks with different depths and configurations. Such extensions will afford improved generalization to networks for extracting different kinds of information.

BIBLIOGRAPHY

[1] O. Aghazadeh, J. Sullivan, and S. Carlsson. Novelty detection from an egocentric perspective. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3297–3304. IEEE, 2011. 30

[2] Forest Agostinelli, Michael R Anderson, and Honglak Lee. Adaptive multi-column deep neural networks with application to robust image denoising. In *Advances in Neural Information Processing Systems 26*, pages 1493–1501, 2013. 87, 94

[3] Ling Bao and Stephen S Intille. Activity recognition from user-annotated acceleration data. In *Pervasive Computing*, pages 1–17. Springer, 2004. 74

[4] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Mach. Learn.*, 36(1-2):105–139, July 1999. 87

[5] Michael Blighe, Hervé Le Borgne, Noel E O'Connor, Alan F Smeaton, and Gareth JF Jones. Exploiting context information to aid landmark detection in sensecam images. In *International Workshop on Exploiting Context Histories in Smart Environments (ECHISE 2006)*, 2006. 76

[6] Steve Branson, Grant Van Horn, Serge Belongie, and Pietro Perona. Bird species categorization using pose normalized deep convolutional nets. *arXiv preprint arXiv:1406.2952*, 2014. 99

[7] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984. 86

[8] Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, August 1996. 87

[9] A. Bulling, J. Ward, H. Gellersen, and G. Tröster. Robust recognition of reading activity in transit using wearable electrooculography. *Pervasive Computing*, pages 19–37, 2008. 10

[10] D. Byrne, A.R. Doherty, C.G.M. Snoek, G.J.F. Jones, and A.F. Smeaton. Everyday concept detection in visual lifelogs: validation, relationships and trends. *Multimedia Tools and Applications*, 49(1):119–144, 2010. 29, 30

[11] Daragh Byrne, Barry Lavelle, Aiden R Doherty, Gareth JF Jones, and Alan F Smeaton. Using bluetooth and gps metadata to measure event similarity in sensecam images. In *IMAI 2007*, 2007. 76

[12] Yuning Chai, Victor Lempitsky, and Andrew Zisserman. Symbiotic segmentation and part localization for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 321–328, 2013. 99

[13] J. Choi, W.J. Jeon, and S.C. Lee. Spatio-temporal pyramid matching for sports videos. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 291–297. ACM, 2008. 23

[14] Dan Ciresan, Ueli Meier, and Jurgen and Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3642–3649, Washington, DC, USA, 2012. IEEE Computer Society. 87, 94

[15] D.C. Ciresan, U. Meier, L.M. Gambardella, and J. Schmidhuber. Convolutional neural network committees for handwritten character classification. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1135–1139, Sept 2011. 87

[16] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, volume 1. IEEE, 2005. 43

[17] Dima Damen, Andrew Gee, Walterio Mayol-Cuevas, and Andrew Calway. Egocentric real-time workspace monitoring using an rgb-d camera. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1029–1036. IEEE, 2012. 55

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 33, 56, 73

[19] Thomas Deselaers, Bogdan Alexe, and Vittorio Ferrari. Localizing objects while learning their appearance. In *ECCV*. Springer, 2010. 37

[20] Michael S Devyver, Akihiro Tsukada, and Takeo Kanade. A wearable device for first person vision(ficcdat workshop). In *3rd International Symposium on Quality of Life Technology*, July 2011. 16, 17

[21] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei A Efros. What makes paris look like paris? *ACM Transactions on Graphics*, 31(4), 2012. 38, 44

[22] Aiden R Doherty, Niamh Caprani, Ciarán Ó Conaire, Vaiva Kalnikaite, Cathal Gurrin, Alan F Smeaton, and Noel E OConnor. Passively recognising human activities through lifelogging. *Computers in Human Behavior*, 27(5):1948–1958, 2011. 76

[23] Aiden R Doherty and Alan F Smeaton. Automatically segmenting lifelog data into events. In *Image Analysis for Multimedia Interactive Services, 2008. WIAMIS'08. Ninth International Workshop on*, pages 20–23. IEEE, 2008. 29

[24] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):743–761, 2012. 12

[25] Piotr Dollár. Piotr's Image and Video Matlab Toolbox (PMT). http://vision.ucsd.edu/ pdollar/toolbox/doc/index.html. 49, 50

[26] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013. 56, 63, 75, 76, 78

[27] Harris Drucker, Corinna Cortes, L. D. Jackel, Yann LeCun, and Vladimir Vapnik. Boosting and other ensemble methods. *Neural Comput.*, 6(6):1289–1301, November 1994. 87

[28] Harris Drucker, Robert E. Schapire, and Patrice Simard. Improving performance in neural networks using a boosting algorithm. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, pages 42–49, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. 87

[29] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008. 63, 78, 81

[30] A. Fathi, Y. Li, and J.M. Rehg. Learning to recognize daily actions using gaze. In *ECCV*, 2012. 9, 20, 26

[31] A. Fathi and J.M. Rehg. Social interactions: A first-person perspective. In *CVPR*, page 8, 2012. 21

[32] A. Fathi, X. Ren, and J.M. Rehg. Learning to recognize objects in egocentric activities. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3281–3288. IEEE, 2011. 13, 15, 19, 26, 37

[33] Alireza Fathi, Ali Farhadi, and James M Rehg. Understanding egocentric activities. In *2011 International Conference on Computer Vision*, pages 407–414. IEEE, 2011. 37

[34] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010. 12, 23

[35] A. Flint, I. Reid, and D. Murray. Learning texton models for real-time scene context. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 41–48. IEEE, 2009. 31, 32

[36] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Comp. and Sys. Science*, 1997. 86

[37] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 1999. 86, 89

[38] Jerome H. Friedman and Of Known (y X)-values. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38:367–378, 1999. 86, 89, 92

[39] Mario Fritz and Bernt Schiele. Decomposition, discovery and detection of visual categories using topic models. In *CVPR*. IEEE, 2008. 37

[40] T. Gao and H. Aghajan. Self lane assignment using egocentric smart mobile camera for intelligent gps navigation. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 57–62. IEEE, 2009. 7

[41] Jim Gemmell, Gordon Bell, Roger Lueder, Steven Drucker, and Curtis Wong. MyLifeBits: fulfilling the memex vision. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 235–238. ACM, 2002. 76

[42] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/ rbg/latent-release5/. 49

[43] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2013. 78

[44] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2013. 85

[45] Philippe-Henri Gosselin, Naila Murray, Hervé Jégou, and Florent Perronnin. Revisiting the fisher vector for fine-grained classification. *Pattern Recognition Letters*, 49:92–98, 2014. 99

[46] P.M. Granitto, P.F. Verdes, and H.A. Ceccatto. Neural network ensembles: evaluation of aggregation algorithms. *Artificial Intelligence*, 163(2):139 – 162, 2005. 87

[47] A. Gupta and L.S. Davis. Objects in action: An approach for combining action understanding and object perception. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007. 25

[48] A. Gupta, A. Kembhavi, and L.S. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(10):1775–1789, 2009. 25

[49] M. Hanheide, N. Hofemann, and G. Sagerer. Action recognition in awearable assistance system. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pages 1254–1258. IEEE, 2006. 25

[50] D.W. Hansen and Q. Ji. In the eye of the beholder: A survey of models for eyes and gaze. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(3):478–500, 2010. 9

[51] M. Havlena, A. Ess, W. Moreau, A. Torii, M. Jancosek, T. Pajdla, and L. Van Gool. Awear 2.0 system: Omni-directional audio-visual data acquisition and processing. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 49–56. IEEE, 2009. 16

[52] D. S. Hayden, C. Vondrick, S. X. Jia, and Y. Landa. The accuracy-obtrusiveness tradeoff for wearable vision platforms. In *IEEE Workshop on Egocentric Vision, CVPR*, volume 1, 2009. xi, 16

[53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 85

[54] S. Hodges, L. Williams, E. Berry, S. Izadi, J. Srinivasan, A. Butler, G. Smyth, N. Kapur, and K. Wood. Sensecam: A retrospective memory aid. *UbiComp 2006: Ubiquitous Computing*, pages 177–193, 2006. 18, 27, 28

[55] Steve Hodges, Emma Berry, and Ken Wood. Sensecam: A wearable camera that stimulates and rehabilitates autobiographical memory. *Memory*, 19(7):685–696, 2011. 54, 75

[56] Forrest Iandola, Matt Moskewicz, Sergey Karayev, Ross Girshick, Trevor Darrell, and Kurt Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014. 85

[57] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(11):1254–1259, 1998. 10

[58] Saharon Rosset Ji Zhu, Hui Zou and Trevor Hastie. Multi-class adaboost. *Statistics and Its Interface*, 2:349–3660, 2009. 88

[59] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 95

[60] T. Judd, K. Ehinger, F. Durand, and A. Torralba. Learning to predict where humans look. In *Computer Vision, 2009 IEEE 12th international conference on*, pages 2106–2113. IEEE, 2009. 10

[61] H. Kang, A.A. Efros, M. Hebert, and T. Kanade. Image matching in large scale indoor environment. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 33–40. IEEE, 2009. 31

[62] Hongwen Kang, Martial Hebert, and Takeo Kanade. Discovering object instances from scenes of daily living. In *ICCV*. IEEE, 2011. 38

[63] Nikolaos Karianakis, Thomas J. Fuchs, and Stefano Soatto. Boosting convolutional features for robust object proposals. *CoRR*, abs/1503.06350, 2015. 87

[64] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 85

[65] Gunhee Kim, Christos Faloutsos, and Martial Hebert. Unsupervised modeling of object categories using link analysis techniques. In *CVPR*. IEEE, 2008. 37

[66] K.M. Kitani, T. Okabe, Y. Sato, and A. Sugimoto. Fast unsupervised ego-action learning for first-person sports videos. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3241–3248. IEEE, 2011. 25

[67] Jonathan Krause, Hailin Jin, Jianchao Yang, and Li Fei-Fei. Fine-grained recognition without part annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5546–5555, 2015. 99

[68] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the 2013 IEEE International Conference on Computer Vision Workshops*, ICCVW '13, pages 554–561, Washington, DC, USA, 2013. IEEE Computer Society. 98, 100

[69] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Dept. of Computer Science, 2009. 95

[70] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012. 56, 75, 78

[71] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 85, 87, 94

[72] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 14

[73] I. Laptev and P. Pérez. Retrieving actions in movies. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007. 14

[74] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE, 2006. 12

[75] Y.J. Lee, J. Ghosh, and K. Grauman. Discovering important people and objects for egocentric video summarization. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1346–1353. IEEE, 2012. 29

[76] Yong Jae Lee, Joydeep Ghosh, and Kristen Grauman. Discovering important people and objects for egocentric video summarization. In *CVPR*, volume 1, pages 3–2, 2012. 55

[77] J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford. A hybrid discriminative/generative approach for modeling human activities. In *Proceedings of the 19th international joint conference on Artificial intelligence*, pages 766–772, 2005. 25

[78] Li-Jia Li, Hao Su, Eric P Xing, and Fei-Fei Li. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, volume 2, page 5, 2010. 75

[79] Yin Li, Alireza Fathi, and James M Rehg. Learning to predict gaze in egocentric video. In *ICCV*, 2013. 55

[80] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999. 65, 77

[81] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 12

[82] Zheng Lu and Kristen Grauman. Story-driven summarization for egocentric video. 2013. 29

[83] et al. M. Douze. Evaluation of gist descriptors for web-scale image search. In *International Conference on Image and Video Retrieval*, 2009. 64

[84] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew B. Blaschko, and Andrea

Vedaldi. Fine-grained visual classification of aircraft. *CoRR*, abs/1306.5151, 2013. 98, 100

[85] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2929–2936. IEEE, 2009. 14

[86] Walterio W Mayol, Andrew J Davison, Ben J Tordoff, ND Molton, and David W Murray. Interaction between hand and wearable camera in 2d and 3d environments. In *Proc. British Machine Vision Conference*, 2004. 55

[87] WW Mayol and DW Murray. Wearable hand activity recognition for event summarization. In *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on*, pages 122–129. IEEE, 2005. 15

[88] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997. 86

[89] Indraneel Mukherjee and Robert E. Schapire. A theory of multiclass boosting. In *NIPS*, 2010. 88

[90] Sobhan Naderi Parizi, Alireza Tavakoli Targhi, Omid Aghazadeh, and Jan-Olof Eklundh. Reading street signs using a generic structured object detection and signature recognition approach. In *VISAPP 2009: Proceedings Of The Fourth International Conference On Computer Vision Theory And Applications, Vol 2*, pages 346–355. Insticc-Inst Syst Technologies Information Control & Communication, 2009. xi, 31, 32

[91] Oliver Nina, Carlos Rubiano, and Mubarak Shah. Action recognition using ensemble of deep convolutional neural networks. 85

[92] B. Noris, M. Barker, J. Nadel, F. Hentsch, F. Ansermet, and A. Billard. Measuring gaze of children with autism spectrum disorders in naturalistic interactions. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pages 5356–5359. IEEE, 2011. 27

[93] D. Mitzel O. Hosseini Jafari and Bastian Leibe. Real-time rgb-d based people detection and tracking for mobile robots and head-worn cameras. In *IEEE International Conference on Robotics and Automation*. IEEE, 2014. 55

[94] US Department of Health and Human Services. 2008 physical activity guidelines for americans. *Be active, healthy, and happy*, 2008. 73

[95] K. Ogaki, K.M. Kitani, Y. Sugano, and Y. Sato. Coupling eye-motion and ego-motion features for first-person activity recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 1–7. IEEE, 2012. 9, 26

[96] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001. 60, 77

[97] Neville Owen, Geneviève N Healy, Charles E Matthews, and David W Dunstan. Too much sitting: the population-health science of sedentary behavior. *Exercise and sport sciences reviews*, 38(3):105, 2010. 73

[98] Kieron OHara, Mischa M Tuffield, and Nigel Shadbolt. Lifelogging: Privacy and empowerment with memories for life. *Identity in the Information Society*, 1(1):155–172, 2008. 55

[99] D.J. Patterson, D. Fox, H. Kautz, and M. Philipose. Fine-grained activity recognition by aggregating abstract object usage. In *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on*, pages 44–51. IEEE, 2005. 11

[100] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *Computer Vision–ECCV 2010*, pages 143–156. Springer, 2010. 77

[101] M. Philipose, K.P. Fishkin, M. Perkowitz, D.J. Patterson, D. Fox, H. Kautz, and D. Hahnel. Inferring activities from interactions with objects. *Pervasive Computing, IEEE*, 3(4):50–57, 2004. 11

[102] Son Lam Phung, Abdesselam Bouzerdoum, and D Chai Sr. Skin segmentation using color pixel classification: analysis and comparison. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(1):148–154, 2005. 56

[103] L. Piccardi, B. Noris, O. Barbey, A. Billard, G. Schiavone, F. Keller, and C. von Hofsten. Wearcam: A head mounted wireless camera for monitoring gaze attention and for the diagnosis of developmental disorders in young children. In *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*, pages 594–598. IEEE, 2007. 16, 17

[104] Hamed Pirsiavash and Deva Ramanan. Detecting activities of daily living in first-person camera views. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2847–2854. IEEE, 2012. xi, 20, 23, 24, 37, 76

[105] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986. 86

[106] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993. 86

[107] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. *arXiv preprint arXiv:1403.6382*, 2014. 56, 63

[108] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015. 85

[109] X. Ren and C. Gu. Figure-ground segmentation improves handled object recognition in egocentric video. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3137–3144. IEEE, 2010. 13

[110] John G Rogers, Alexander JB Trevor, Carlos Nieto-Granda, and Henrik I Christensen. Simultaneous localization and mapping with learned object recognition and semantic data association. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1264–1270. IEEE, 2011. 31

[111] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *arXiv preprint arXiv:1409.0575*, 2014. 85

[112] Bryan C Russell, William T Freeman, Alexei A Efros, Josef Sivic, and Andrew Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, volume 2. IEEE, 2006. 37

[113] Michael S. Ryoo and Larry Matthies. First-person activity recognition: What are they doing to me? In *CVPR*, pages 2730–2737. IEEE, 2013. 55

[114] M. Saberian and N. Vasconcelos. Multiclass boosting: Theory and algorithms. In *NIPS*, 2011. 88

[115] Jorge Sánchez and Florent Perronnin. High-dimensional signature compression for large-scale image classification. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1665–1672. IEEE, 2011. 77

[116] B. Schiele, N. Oliver, T. Jebara, and A. Pentland. An interactive computer vision system dypers: Dynamic personal enhanced reality system. *Computer Vision Systems*, pages 51–65, 1999. 13

[117] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 32–36. IEEE, 2004. 14

[118] Holger Schwenk. The diabolo classifier. *Neural Computation*, 10(8):2175–2200, 1998. 87

[119] Holger Schwenk and Yoshua Bengio. Adaboosting neural networks: Application to on-line character recognition. In *Artificial Neural Networks - ICANN '97, 7th International Conference, Lausanne, Switzerland, October 8-10, 1997, Proceedings*, pages 967–972, 1997. 87

[120] Holger Schwenk and Yoshua Bengio. Boosting neural networks. *Neural Computation*, 12(8):1869–1887, 2000. 87, 91

[121] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 85, 87, 94

[122] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 85

[123] Saurabh Singh, Abhinav Gupta, and Alexei A Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV*. Springer, 2012. 38, 44

[124] Josef Sivic, Bryan C Russell, Alexei A Efros, Andrew Zisserman, and William T Freeman. Discovering object categories in image collections. 2005. 37

[125] E.H. Spriggs, F. De La Torre, and M. Hebert. Temporal segmentation and activity classification from first-person sensing. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 17–24. IEEE, 2009. 11

[126] Ju Sun, Xiao Wu, Shuicheng Yan, Loong-Fah Cheong, T.-S. Chua, and Jintao Li. Hierarchical spatio-temporal context modeling for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition.*, 2009. 56

[127] L. Sun, U. Klank, and M. Beetz. Eyewatchme3d hand and object tracking for

inside out activity analysis. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 9–16. IEEE, 2009. 9, 15

[128] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014. 85

[129] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1701–1708. IEEE, 2014. 85

[130] Antonio Torralba, Kevin P. Murphy, William T. Freeman, and Mark A. Rubin. Context-based vision system for place and object recognition. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, 2003. 56

[131] Aruni RoyChowdhury Tsung-Yu Lin and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *International Conference on Computer Vision (ICCV)*, 2015. 85, 98, 99

[132] Michael Van den Bergh, Xavier Boix, Gemma Roig, Benjamin de Capitani, and Luc Van Gool. Seeds: Superpixels extracted via energy-driven sampling. In *Computer Vision–ECCV 2012*, pages 13–26. Springer, 2012. 61

[133] Vladimir Vezhnevets, Vassili Sazonov, and Alla Andreeva. A survey on pixel-based skin color detection techniques. In *Proc. Graphicon*, volume 3, pages 85–92. Moscow, Russia, 2003. 56, 60

[134] P. Viola and M.J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004. 12

[135] Paul Viola and Michael Jones. Robust real-time object detection. *Workshop on Statistical and Computational Theories of Vision*, 2001. 86

[136] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011. 97, 100

[137] Markus Weber, Max Welling, and Pietro Perona. Towards automatic discovery of object categories. In *PAMI*, volume 2, pages 101–108. IEEE, 2000. 37

[138] Jianxin Wu, A. Osuntogun, T. Choudhury, M. Philipose, and J.M. Rehg. A scalable approach to activity recognition based on object use. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1 –8, oct. 2007. 11

[139] Jieping Ye, Zheng Zhao, and Mingrui Wu. Discriminative k-means for clustering. *NIPS*, 20, 2007. 38

[140] Z. Ye, Y. Li, A. Fathi, Y. Han, A. Rozga, G.D. Abowd, and J.M. Rehg. Detecting eye contact using wearable eye-tracking glasses. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 699–704. ACM, 2012. 9, 27

[141] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional neural networks. *arXiv preprint arXiv:1311.2901*, 2013. 75

[142] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional neural networks. *arXiv preprint arXiv:1311.2901*, 2013. 85

[143] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In *Computer Vision–ECCV 2014*, pages 834–849. Springer, 2014. 99

[144] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(12):239 – 263, 2002. 87